Introduction to Stochastic Petri Nets

Gianfranco Balbo

Università di Torino, Torino, Italy,
Dipartimento di Informatica
`balbo@di.unito.it`

Abstract. Stochastic Petri Nets are a modelling formalism that can be conveniently used for the analysis of complex models of Discrete Event Dynami Systems (DEDS) and for their performance and reliability evaluation. The automatic construction of the probabilistic models that underly the dynamic behaviours of these nets rely on a set of results that derive from the theory of untimed Petri nets. The paper introduces the basic motivations for modelling DEDS and briefly overviews the basic results of net theory that are useful for the definition of Stochastic Petri Nets and Generalized Stochastic Petri Nets. The different approaches that have been used for introducing the concept of time in these models are discussed in order to provide the basis for the definition of SPNs and GSPNs as well. Details on the solution techniques and on ntheir computational aspects are provided. A brief overview of more advanced material is included at the end of the paper to highlight the state of the art in this field and to give pointers to relevant results published in the literature.

1 Introduction

Petri nets [60,1,59,63] are a powerful tool for the description and the analysis of systems that exhibit concurrency, synchronization and conflicts. Timed Petri nets [7,54] in which the basic model is augmented with time specifications are commonly used to evaluate the performance and reliability of complex systems.

The pioneering work in the area of timed Petri nets was performed by Merlin and Faber [50], and by Noe and Nutt [58]. In this early work, timed Petri nets were viewed as a formalism for the description of the global behaviour of complex structures. The nets were used to *tell* all the possible stories that systems could experience based on their temporal specifications and analysis was conducted on the basis of observations made on these stories.

Following these initial ideas, several proposals for incorporating timing information into Petri net models appeared in the literature. Interpreting Petri nets as state/event models, time is naturally associated with activities that induce state changes, and hence with the delays incurred before firing transitions.

Stochastic Petri Nets (SPNs) were introduced in 1980 [68,56,53] as a formalism for the description of Discrete Event Dynamic Systems (DEDS) whose

E. Brinksma, H. Hermanns, and J.-P. Katoen (Eds.): FMPA 2000, LNCS 2090, pp. 84–155, 2001.

dynamic behaviour could be represented by means of continuous-time homogeneous Markov chains. The original SPN proposal assumed atomic firings, exponentially distributed firing times, and a race execution policy; i.e., when multiple transitions are simultaneously enabled, the race policy selects the transition with the statistically minimum delay to fire.

With the aim of extending the modelling power of stochastic Petri nets, Generalized Stochastic Petri Nets (GSPNs) were proposed in [4]. GSPNs include two classes of transitions: exponentially distributed *timed* transitions, which are used to model the random delays associated with the execution of activities, and *immediate* transitions, which are devoted to the representation of logical actions that do not consume time. Immediate transitions permit the introduction of branching probabilities that are independent of timing specifications. When timed and immediate transitions are enabled in the same marking, immediate transitions always fire first. In GSPNs the reachability set is also partitioned in two sets. *Tangible* markings are those in which only timed transitions are enabled whereas *vanishing* markings are those in which at least one immediate transition is enabled. The time spent by a GSPN in a tangible state is exponentially distributed with the parameter depending on the timed transitions that are enabled in that marking; the time spent by a GSPN in a vanishing marking is instead zero. Other generalizations of the basic SPN formalism that are related to GSPNs, are the Extended Stochastic Petri Nets [35] and the Stochastic Activity Networks [51]. GSPNs are among the SPN formalisms that are most commonly used for the analysis of important problems and a considerable effort has been devoted to their improvement since the time of their original introduction.

In this paper, we discuss the relevance of this modelling formalism by providing first a broad view of its application field and by introducing the basic results that set the ground for the derivation of the stochastic processes corresponding to these models and for the study of their solution methods. The balance of the paper is the following. Section 2 briefly discusses the relevance of modelling to support the analysis and the design of complex systems. Section 3 introduces the characteristics of Discrete Event Dynamic Systems and discusses the role that models play in the analysis and the design of applications that can be represented within this framework. Section 4 describes the relevance of Petri nets for modelling Discrete Event Dynamic Systems and introduces the basic classical properties of the formalism that are needed later in the paper. Section 5 briefly surveys the impact that a global priority structure has on the properties and the behaviours of Petri nets. Section 6 presents the different possibilities that exist for introducing the concept of time in Petri net models. Section 7 introduces the definition of Stochastic Petri nets and provides the details for the construction of their underlying stochastic process. Section 8 discusses the characteristics of Generalized Stochastic Petri Nets and provides details on some of the computational issues that are relevant for the application of this modeling formalism. Section 9 concludes the paper with a few pointers to more advanced material. and with some general remarks on net modelling.

The paper has been written in the attempt of providing a uniform and self-contained introduction to the problem of modelling Discrete Event Dynamic Systems with Stochastic Petri Nets. The discussion introduces the basic terminology and operational rules of Petri nets for which a comprehensive reference can be found in [55] where the reader will be able to find a clear explanation of all the concepts that are only marginally addressed in this work.

2 Models

Understanding the behaviour of real systems is always difficult due to the complexity of their organization and to the intricacy of the interactions among their components.

Designing and managing real systems often require that relationships between organizational choices and attained results be identified in order to decide on possible improvements of the system architecture and of the operational environment for achieving better performance.

In all these cases, reasoning on the behaviour of systems can become more reliable if proper descriptions are available that help in clarifying the relationships among system components. The best way of obtaining such descriptions is that of constructing a model that highlights the important features of the system organization and provides ways of quantifying its properties neglecting all those details that are relevant for the actual implementation, but that are marginal for the objective of the study.

Models can be developed for a variety of reasons that include understanding and learning about the behaviour of the system, improving its performance and making decisions about its design or its operation. Models are useful for explaining why and how certain features of the system actually occur. The model helps in developing insights on the operation of the system and in understanding the directions of the influence that certain input parameters may have on the results.

The mathematical formulation of a model provides ways for formal reasoning about the behaviour of a real system in a manner that is safe (although difficult in same cases) and that is amenable for automatization.

The possibility of computing results from the analysis of a model is the key for closing a loop that starts from the abstraction of the relevant features of the system during modelling construction and that ends with the interpretation of the results provided by the model and reflected on the real system.

3 Discrete Event Dynamic Systems

A system is often defined as a collection of *objects* together with their *relations*. Objects are characterized by *attributes* some of which are fixed, while others are variable. The value assumed by a variable attribute contributes to the definition of the state of the object (*local* state). The *state* of the system (*global* state) results from the composition of the states of the individual objects (*components*). The relations among objects represent the constraints that drive the change of states. Abstracting from the particular event that may cause the change of state, we call *transitions* such change of state patterns. We can say that state variables are *passive* elements, while transitions are *active*.

Discrete Event Dynamic Systems (DEDS) are systems with a discrete *state space* (i.e., they have a countable - possibly infinite - number of states) and whose evolution is not directly due to the passage of time, but to the occurrence of *events*. Depending on whether events happen at arbitrary points in time or only at precise instants, DEDS are called *asynchronous* or *synchronous*

3.1 DEDS as a View

Many real systems can be viewed as DEDS not because of their intrinsic characteristics, but because of the aspects of their behaviour that we want to emphasize. For instance, a water reservoir that contains a continuously varying quantity of water, can be viewed as a DEDS if we restrict our attention to the fact that the water exceeds or not a predefined safety level. In this case the reservoir can assume only two states (safe and un-safe) and the events that may cause the change of state can be identified in the occurrence of thunderstorms and in the openings of the dam.

We thus always speak of DEDS systems referring either to DEDS view of a real system or to a system that can be viewed in this way.

DEDS systems can be identified within quite different application domains. In Flexible Manufacturing, the state of the system may be represented by the number of parts present in front of the different machine-tools and the events may correspond to the completions of the activities performed by different machine-tools, to the production of a good, or to the arrival of new raw parts

In Computing, the state of the system may corresponds to the number of tasks currently in process as well as to those waiting for the completion of some I/O actions; examples of events are in this case the CPU quantum expiration, the interrupts coming from the I/O devices and the traps due to system call executions. In Telecommunication, the state may corresponds to the number of packets stored in the different buffers and the events to message submissions as well as to protocol actions. Finally, in Traffic the state of the system may corresponds to the number of cars waiting in a parking lot, or at a crossroad, or using a section of road while events may correspond to arrival and departures of cars as well as to changes of semaphore colours.

In all these cases, independently of the actual meaning of the different components, understanding the behaviour of these system is usually hard because of the intrinsic complexity of the problem (e.g., the number of machine tools and of parts of Flexible Manufacturing System may give rise to millions of states that may be difficult to envision and to keep under control), because of the subtle interference among components (e.g., in Traffic systems the existence of bottlenecks, temporarily hidden by the presence of other congestion points that generate huge traffic-jams, may defeat the expected improvements coming from the - often costly - removal of such delay causes), and of the paradoxical relationship among the operations of certain components (e.g., in Time Sharing Computing System increasing the multiprogramming level to keep the CPU busy may yield an actual reduction of the CPU utilization because of the consequent increasing of the page fault rate - thrashing effect). Reasoning about the behaviour of such systems is difficult without the support of proper models.

Formal models are thus the basis for a better understanding of existent systems and for the effective and efficient design of new solutions.

3.2 Performance Evaluation of Discrete Event Dynamic Systems

One of the reasons for modeling DEDS is that of constructing a formal representation that can be used to drive design decisions toward efficient solutions and to optimize system operation to obtain the best results at the minimum cost.

As real systems may undergo failures, DEDS models must be capable of representing such failure/repair cycles in order to design real systems that perform well also in presence of temporary failures. Performance and Dependability Evaluation is the discipline that uses mathematical and simulation models for the computation of time-related performance indices such as resource utilization, system productivity and system response time accounting for system failures.

To compute these results the modelling formalism must include the possibility for time specifications (usually expressed in terms of the delay needed for performing a given action) and for routing/selection information.

The great diversity of real systems is commonly reflected at the level of time and selection specifications by means of random variables, thus leading to the construction and the solution of probabilistic models. Performance indices are evaluated for DEDS in these cases through the computation of the probability of finding the DEDS in each of its states.

4 Petri Net Models of DEDS

Petri nets (PNs) are abstract formal models that have been developed in search for natural, simple, and powerful methods for describing and analyzing the flow of information and control in systems. Petri nets have been originally proposed for the description and the analysis of systems in which concurrency and conflicts play a special role. In Petri nets, the state of the system derives from the combination of local state variables that allow a direct representation of concurrency, causality, and independence. Petri nets are graphically represented as collections of places and transitions connected by directed arcs so to form bi-partite graphs. The connections of transitions with places by means of input and output arcs represents the pre- and post-conditions, respectively, for the transitions to be enabled to fire. These conditions can be captured by the *incidence* matrix of the model that is the basis for the computation of a large set of *structural* results that represent the real advantage of using these type of models. The graphical aspect of these models are very attractive for practical modelling since they help in understanding how features of the real system are conveyed in the model.

In order to keep the PN models of DEDS concise, high level PN have been introduced that provide a form of abbreviation when repetition of similar subnets would make the model large and difficult to understand. Always to make models easier to understand, several extensions have been introduced in the basic PN formalism, often with the disadvantage of reducing the *analyzability* of the model (e.g., inhibitor arcs give to the Petri net formalism the computation power of Turing Machines, but their effect is, in general, neglected during the structural analysis of the net).

The behaviour of PNs is independent of time and environment and is characterized by the *non-deterministic* firing of transitions that are simultaneously enabled in a given marking. The connection of the formalism with reality is provided in this case by *interpretations* that incorporate in the model external constraints such as time considerations. Different extensions and different interpretations yield different PN based formalisms sharing some basic principles.

Petri nets are models consisting of two parts:

1. A net structure - an inscribed bipartite directed graph, that represents the static part of the system. The two types of nodes are called places and transitions and are represented as circles and boxes (or bars), respectively. Places correspond to state variables of the system and transitions to actions that induce changes of states. Arcs connecting places to transitions are called *input* arcs; *output* arcs connect instead transitions to places. Different types of inscriptions lead to various families of nets. When the inscriptions are natural numbers associated with arcs, named weights or multiplicities, Place/Transition (P/T) nets are obtained.
2. A marking - an assignment of tokens to places. The marking of a place represents its state value.

The specification of a PN model is completed by the definition of an initial marking. The dynamics of a system (i.e., its behaviour) is given by the evolution of the marking that is driven by few simple rules. The basic rule allows the occurrence of a transition when the input state values fulfill some conditions expressed by the arc inscriptions. The occurrence of a transition changes the values of its adjacent state variables (markings of input and output places) according to arc inscriptions again. This separation allows to reason on net based models at two different levels: *structural* and *behavioural.* From the former we may derive "fast" conclusions on the possible behaviours of the modelled system. Pure behavioural reasonings can be more conclusive, but they may require substantial computations, which in certain cases may not even be feasible. Structural reasoning may be regarded as an *abstraction* of the behavioural one: for instance, instead of studying whether a given system has a finite state space, we might address the problem of whether the state space is finite *for every* possible initial state; similarly, we could investigate whether *there exists* an initial marking that guarantees infinite activity, rather than verifying if this is the case for a given initial state.

A marked Petri net is formally defined by the following tuple

$$PN = (P,\ T,\ F,\ W,\ \boldsymbol{m}_0)$$

where
$P = (p_1,\ p_2,\ ...,\ p_P)$ is the set of places,
$T = (t_1,\ t_2,\ ...,\ t_T)$ is the set of transitions,
$F \subseteq (P \times T) \bigcup (T \times P)$ is the set of arcs,
$W : F \rightarrow \mathbb{N}$ is a weight function,
$\boldsymbol{m}_0 = (m_{01},\ m_{02},\ ...,\ m_{0P})$ is the initial marking.

As we have said, a marking $\boldsymbol{m}$ is an assignment of tokens to places and can thus be represented by a vector with as many components as there are places in the net: the $i-th$ component of such a vector represents the number of tokens assigned to place p_i. When nets are large, a more convenient notation for markings is that of expressing the assignment of tokens by means of a formal sum in which we explicitly represent the name of a marked place multiplied by the number of tokens assigned to it. A marking that has h tokens in place i and k tokens in place j (only) will be denoted as $\boldsymbol{m} = hp_i + kp_j$ (a formal sum denoting the multiset on P defined by the marking).

The dot notation is used for pre- and post-sets of nodes: ${}^\bullet v = \{u| < u, v > \in F\}$ and $v^\bullet = \{u| < v, u > \in F\}$. A pair comprising a place p and a transition t is called a *self-loop* if p is both input and output of t ($p \in {}^\bullet t \bigwedge p \in t^\bullet$ - see Figure 1).

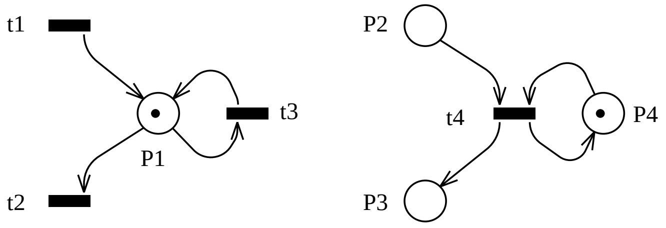


Fig. 1. Self-loops

A net is said to be pure if it has no self-loops. Pure nets are completely characterised by a single matrix $\boldsymbol{C}$ that is called the *incidence matrix* of the net and that combines the information provided by the flow relations and by the weight function.

$$\boldsymbol{C} = \begin{array}{c|c} & \textit{transitions} \\ \hline \textit{places} & \boxed{\quad c_{pt} \quad} \end{array}$$

with $c_{pt} = c^{+}_{pt} + c^{-}_{pt} = w(t,p) - w(p,t)$

4.1 System Dynamics

The graph and matrix characterizations that we have described in the previous section represent the static component of a PN model. The dynamic evolution of the PN marking is governed by transition occurrences (firings) which consume and create tokens.

"Enabling" and "firing" rules are associated with transitions. The enabling rule states the conditions under which transitions are allowed to fire. The firing rule defines the marking modification induced by the occurrence of a transition. Informally, we can say that the enabling rule defines the *conditions* that allow a transition to fire, and the firing rule specifies the *change* of state produced by the transition.

Both the enabling and the firing rules are specified in terms of arc characteristics. In particular, the enabling rule involves (most of the time) input arcs only, while the firing rule depends on input and output arcs. Note that input arcs play a double role, since they are involved both in enabling and in firing.

A transition t is *enabled* if and only if each input place contains a number of tokens *greater or equal* than given thresholds defined by the multiplicities of arcs. Formally, this condition is expressed by the following definition:

Definition 1 (Enabling). *Transition t is enabled in marking $\boldsymbol{m}$ if and only if*

- $\forall p \in {}^{\bullet}t, m(p) \geq O(t,p)$
  that, in matrix notation is equivalent to
- $\boldsymbol{m} \geq \boldsymbol{c}(.,t)^{-T}$

The set of transitions enabled in marking $\boldsymbol{m}$ is indicated with $E(\boldsymbol{m})$; the number of simultaneous enablings of a transition t_i in a given marking $\boldsymbol{m}$ is called its enabling degree, and is denoted by $e_i(\boldsymbol{m})$.

When transition t fires, it *deletes* from each place in its input set ${}^{\bullet}t$ as many tokens as the multiplicity of the arc connecting that place to t, and *adds* to each place in its output set $t^{\bullet}$ as many tokens as the multiplicity of the arc connecting t to that place.

Definition 2 (Firing). *The firing of transition t, enabled in marking $\boldsymbol{m}$, produces marking $\boldsymbol{m}'$ such that*

- $\boldsymbol{m}' \;=\; \boldsymbol{m} \;+\; O(t) \;-\; I(t)$
  again, this same relation is expressed in matrix notation as follows
- $\boldsymbol{m}' \;=\; \boldsymbol{m} \;-\; \boldsymbol{c}(.,t)^{-T} \;+\; \boldsymbol{c}(.,t)^{+T}$

This statement is usually indicated in a compact way as $\boldsymbol{m}[t\rangle\boldsymbol{m}'$, and we say that $\boldsymbol{m}'$ is *directly reachable* from $\boldsymbol{m}$.

The natural extension of the concept of transition firing, is the firing of a *transition sequence* (or execution sequence). A transition sequence[1] $\sigma = t_{(1)}, \cdots, t_{(k)}$ can fire starting from marking $\boldsymbol{m}$ if and only if there exists a sequence of markings $\boldsymbol{m} = m_{(1)}, \cdots, m_{(k+1)} = \boldsymbol{m}'$ such that $\forall i = (1, \cdots, k), \boldsymbol{m}_{(i)}[t_{(i)}\rangle\boldsymbol{m}_{(i+1)}$. We denote by $\boldsymbol{m}[\sigma\rangle\boldsymbol{m}'$ the firing of a transition sequence, and we say that $\boldsymbol{m}'$ is *reachable* from $\boldsymbol{m}$.

An important final consideration is that the enabling and firing rules for a generic transition t are "local": indeed, only the numbers of tokens in the input of t, and the weights of the arcs connected to t need to be considered to establish whether t can fire and to compute the change of marking induced by the firing of t. This justifies the assertion that the PN marking is intrinsically "distributed".

A common way of describing the behaviour of a P/T system is by means of its sequential observation. A hypothetical observer is supposed to "see" only single events occurring at any point in time. The interleaving semantics of a net system is given by all possible sequences of individual transition firings that could be observed from the initial marking. If two transitions t_1 and t_2 are enabled simultaneously, and the occurrence of one does not disable the other, in principle they could occur at the same time, but the sequential observer will see either t_1 followed by t_2 or viceversa. The name interleaving semantics comes from this way of seing simultaneous occurrences.

Starting from the initial marking it is possible to compute the set of all markings reachable from it (the state space of the PN) and all the paths that the system may follow to move from state to state.[2]

Definition 3. *The Reachability Set of a PN with initial marking $\boldsymbol{m}_0$ is denoted $RS(\boldsymbol{m}_0)$, and is defined as the smallest set of markings such that*

- $\boldsymbol{m}_0 \in RS(\boldsymbol{m}_0)$
- $\boldsymbol{m}_1 \in RS(\boldsymbol{m}_0) \wedge \exists t \in T : \boldsymbol{m}_1[t\rangle\boldsymbol{m}_2 \Rightarrow \boldsymbol{m}_2 \in RS(\boldsymbol{m}_0)$

[1] We write $t_{(1)}$ rather than t_1 because we want to indicate the first transition in the sequence, that may not be the one named t_1.

[2] Obviously this computation is feasible only in the case of models with finite state spaces. In the rest of this paper, we assume that our models satisfy this condition, except when it is sted differently. Proper generalisations are possible to deal with infinite state spaces introducing the notion of "covering tree" [55].

When there is no possibility of confusion we indicate with RS the set $RS(\boldsymbol{m}_0)$. We also indicate with $RS(\boldsymbol{m})$ the set of markings reachable from a generic marking $\boldsymbol{m}$.

The RS contains no information about the transition sequences fired to reach each marking. This information is contained in the reachability graph, where each node represents a reachable state, and there is an arc from $\boldsymbol{m}_1$ to $\boldsymbol{m}_2$ if the marking $\boldsymbol{m}_2$ is directly reachable from $\boldsymbol{m}_1$. If $\boldsymbol{m}_1[t\rangle\boldsymbol{m}_2$, the arc is labelled with t. Note that more than one arc can connect two nodes (it is indeed possible for two transitions to be enabled in the same marking and to produce the same state change), so that the reachability graph is actually a multigraph.

Definition 4. *Given a PN system, and its reachability set RS, we call Reachability Graph $RG(\boldsymbol{m}_0)$ the labelled directed multigraph whose set of nodes is RS, and whose set of arcs A is defined as follows:*

$$\begin{array}{l} \bullet A \subseteq RS \times RS \times T \\ \bullet \langle \boldsymbol{m}_i, \boldsymbol{m}_j, t\rangle \in A \Leftrightarrow \boldsymbol{m}_i[t\rangle\boldsymbol{m}_j \end{array} \qquad (1)$$

$\boldsymbol{m}_0$ is taken as the initial node of the graph.

Multiple events may happen at any given time. A *step S* is a multi-set of transitions that are enabled to concurrently fire in the same marking. Firing a step amounts to withdraw the tokens from all the input places of the transitions of the step and to deposit tokens in all their output places. If a set of transitions can be fired in a step, this makes explicit the fact that they need not occurring in a precise order. The occurrence of a step can be denoted by $\boldsymbol{m} \xrightarrow{S} \boldsymbol{m}'$, or $\boldsymbol{m} \xrightarrow{\sigma} \boldsymbol{m}'$, if σ is an arbitrary sequentialisation of S. In fact, every sequentialisation of the step is fireable so that, in practice, the reachable markings can be computed considering individual transition occurrences only.

The dynamic behaviour of Petri net models is characterized by three basic phenomena that account for the fact that actions may occur simultaneously (*concurrency*), some actions require that others occur first (*causal dependency*), and actions may occur only in alternative (*conflicts*).

Concurrency - Two transitions are concurrent in a given marking if they can occur in a step.

Definition 5. *Transitions t_i and t_j are in a concurrency relation in marking $\boldsymbol{m}$, denoted by $< t_i, t_j > \in CO(\boldsymbol{m})$, if $\boldsymbol{m} \xrightarrow{t_i} \boldsymbol{m}'$, $\boldsymbol{m} \xrightarrow{t_j} \boldsymbol{m}''$, $e_j(\boldsymbol{m}') > 0$, and $e_i(\boldsymbol{m}'') > 0$.*

in other words, $< t_i, t_j > \in CO(\boldsymbol{m})$, if $\boldsymbol{m} > \boldsymbol{c}(., t_i)^T + \boldsymbol{c}(., t_j)^T$. Notice that steps allow to express true concurrency. In the case of interleaving semantics, as we mentioned before, concurrency of two (or more) actions t_1 and t_2 is represented by the possibility of performing them in any order, first t_1 and then t_2, or viceversa. Nevertheless, the presence of all possible sequentialisations of the actions does not imply that they are "truly" concurrent, as the example in Figure 2 illustrates: t_1 and t_2 can occur in any order, but they cannot occur simultaneously, and in fact the step $t_1 + t_2$ is not enabled. The distinction is especially important if transitions t_1 and t_2 were to be refined, i.e., if they were to be replaced by subnets.

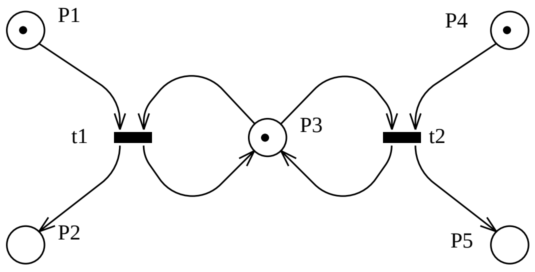


Fig. 2. Shared place

Causal Dependence - Informally, causal dependencies are represented by the partial ordering of actions induced by the flow relation. They correspond to situations in which the firing of a given transition can happen only after the occurrence of others in whatever order.

Definition 6. *Transitions t_i is in direct causality relation with t_j in marking $\boldsymbol{m}$, denoted by $< t_i, t_j > \in DC(\boldsymbol{m})$, if $\boldsymbol{m} \xrightarrow{t_i} \boldsymbol{m}'$, and $e_j(\boldsymbol{m}') > e_j(\boldsymbol{m})$.*

The very basic net construct used to model causal dependences is a place connecting two transitions. Transitions connected through a place are said to be in structural causal connection relation $< t_i, t_j > \in SCC(\boldsymbol{m})$, if $t_i^\bullet \bigwedge {}^\bullet t_j \neq \emptyset$.

Conflicts - Informally, we have a situation of *conflict* when, being several transitions enabled in the same marking, we have to chose which one to fire and, by so doing, we affect the enabling conditions of the others. We can thus say that a transition t_r is in conflict with transition t_s in marking $\boldsymbol{m}$ iff $t_r, t_s \in E(\boldsymbol{m})$, $\boldsymbol{m} \xrightarrow{t_s} \boldsymbol{m}'$, and $t_r \notin E(\boldsymbol{m})$. Things are more complex when we consider concurrent systems where the fact that two transitions are enabled in a given marking does not necessarily means that we have to choose which one to fire even if they share some input places. Formally, we have a situation of conflict when the set of transitions enabled in a given marking is not a step.

Definition 7. *Transition t_i is said to be in effective conflict relation with transition t_j in marking $\boldsymbol{m}$, denoted by $< t_i, t_j > \in EFC(\boldsymbol{m})$, if $\boldsymbol{m} \xrightarrow{t_i} \boldsymbol{m}'$, and $e_j(\boldsymbol{m}') < e_j(\boldsymbol{m})$.*

This relation is antisymmetric as we can see from the second example of Figure 3 where t'_2 is in effective conflict with t'_1, but not the other way around, since the firing of t'_1 does not decrease the enabling degree of t'_2.

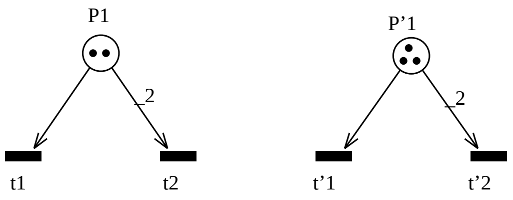


Fig. 3. Effective conflicts

The very basic net construct used to model conflicts is a place with more than one output transition. The output transitions of a place of this type are said to be in *structural conflict relation* ($< t_i, t_j > \in SC$, if $t_i^\bullet \bigwedge {}^\bullet t_j \neq \emptyset$.

This relation is reflexive and symmetric, but not transitive. Its transitive closure is named *coupled conflict relation* and partitions the transitions of a net into *coupled conflict sets. CCS(t)* denotes the coupled conflict set containing t. In Figure 4, transitions t_1 and t_2 are in structural conflict, while transitions t_3 and t_5 are not in structural conflict, but they are in coupled conflict relation, through t_4.

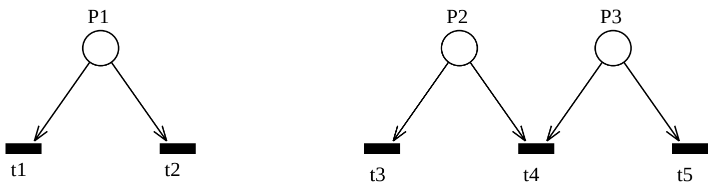


Fig. 4. Structural conflicts

It is important to remark the difference between structural conflicts and effective conflicts which depends on the marking of the net. A structural conflict makes possible the existence of an effective conflict, but does not guarantee it, except in the case of equal conflicts where all the transitions have the same input set.

Definition 8. *Transitions* t_i *and* t_j *are said to be in equal conflict relation, denoted by* $< t_i, t_j > \in EQ$, *if* $^\bullet t_i = {}^\bullet t_j$

Figure 5 shows an equal conflict set.

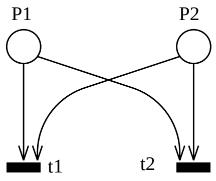


Fig. 5. Equal conflict

When structural conflicts are not equal, it may happen that transitions initially in conflict may subsequently become elements of a step or, viceversa, when the interleaved firing of a step yields to conflict situations. The cases depicted in Figure 6 show situations of this type.

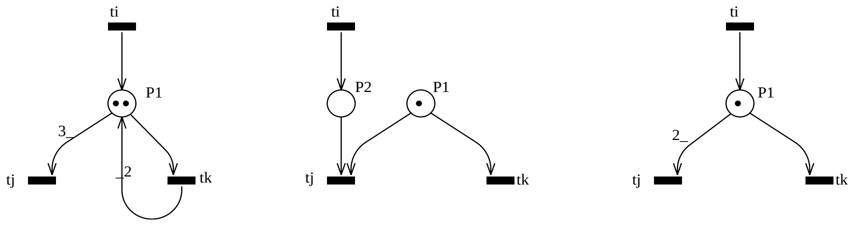


Fig. 6. Non equal conflicts

An intriguing situation arises when different interleaved firings of the members of a step may either yield conflict situations or not. This phenomenon is known as *confusion* and is again depicted by the conflicts of Figure 6, where we can recognize that t_i and t_k are a step such that, when t_k fires no effects are felt by transition t_i, while the same is not true in the other case.

Properties of Petri Nets - Properties of Petri net models are characteristics that allow to assess the quality of a given system in an objective manner. The following are among the most useful properties that can be defined for Petri net models.

Reachability and reversibility — As defined in the previous sections, a marking $\boldsymbol{m}'$ is *reachable* from $\boldsymbol{m}$ if there exists a sequence σ such that $\boldsymbol{m}[\sigma\rangle\boldsymbol{m}'$.

Reachability can be used to answer questions concerning the possibility for the modelled system of being in a given marking $\boldsymbol{m}$. An important reachability property is *reversibility*: a marked Petri net is said to be *reversible* if and only if from any state reachable from $\boldsymbol{m}_0$, it is possible to come back to $\boldsymbol{m}_0$ itself. More formally, a Petri net with initial marking $\boldsymbol{m}_0$ is reversible if and only if $\forall \boldsymbol{m} \in RS(\boldsymbol{m}_0), \boldsymbol{m}_0 \in RS(\boldsymbol{m})$. Reversibility expresses the possibility for a PN to come back infinitely often to its initial marking. In general, we say that $\boldsymbol{m} \in RS(\boldsymbol{m}_0)$ is a *home state* for the PN if and only if $\forall \boldsymbol{m}' \in RS(\boldsymbol{m}_0), \boldsymbol{m} \in RS(\boldsymbol{m}')$. A marking $\boldsymbol{m}_h$ is called a *home-state* iff

$$\forall \boldsymbol{m} \ \in \ RS(\boldsymbol{m}_0), \quad \boldsymbol{m}_h \ \in \ RS(\boldsymbol{m})$$

The set of the home-states of a Petri net is called its *home-space*

A Petri net is *reversible* whenever its initial marking $\boldsymbol{m}_0$ is a home-state

Liveness — A transition t is said to be *live* if and only if, for each marking $\boldsymbol{m}$ reachable from $\boldsymbol{m}_0$, there exists a marking $\boldsymbol{m}'$, reachable from $\boldsymbol{m}$, such that $t \in E(\boldsymbol{m}')$. Formally, transition t_r is *live* iff

$$\forall \boldsymbol{m} \ \in \ RS(\boldsymbol{m}_0), \quad \exists \boldsymbol{m}' \ : \ (\ \boldsymbol{m} \ \overset{s}{\rightarrow} \ \boldsymbol{m}' \bigwedge t_r \ \in \ E(\boldsymbol{m}') \ )$$

A Petri net is said to be live iff $\forall t_r \ \in \ T \ : \ t_r$ is live. Liveness is a property that depends on the initial marking. A transition that is not live is said to be *dead*. For each dead transition t, it is possible to find a marking $\boldsymbol{m}$ such that none of the markings in $RS(\boldsymbol{m})$ enables t.

A very important consequence of liveness is that, if at least one transition is live, then the Petri net cannot deadlock.[3] Moreover, if all transitions are live, then the corresponding Petri net contains no livelock.[4] Liveness defines the possibility for a transition to be enabled (and to fire) infinitely often.

Boundedness — A place p of a Petri net is said to be *k-bounded* if and only if, for each reachable marking $\boldsymbol{m}$, the number of tokens in that place is less than or equal to k. Formally, we have that a place p_i is bounded (k-bounded) iff

$$\forall \boldsymbol{m} \ \in \ RS(\boldsymbol{m}_0), \quad \exists k \ : \ m_i \leq \ k$$

A Petri net is said to be *k-bounded* if and only if all places $p \in P$ are *k-bounded*. Petri nets that are 1-bounded are said to be *safe*. A very important consequence of boundedness is that it implies the finiteness of the state space. In particular, if a Petri net comprising N places is *k-bounded*, the number of states cannot exceed $(k+1)^N$.

[3] A Petri net contains a deadlock if it can reach a state in which no transition can be fired.

[4] A system is in a livelock condition when it enters a subset of its activities from which it has no possibility of exiting.

It is interesting to note that boundedness, liveness and reversibility are (good) independent properties of a Petri net [55].

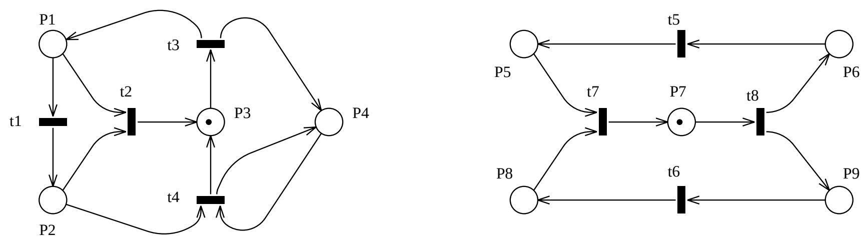


Fig. 7. Examples of two nets that are $\overline{BLR}$ and BLR, respectively.

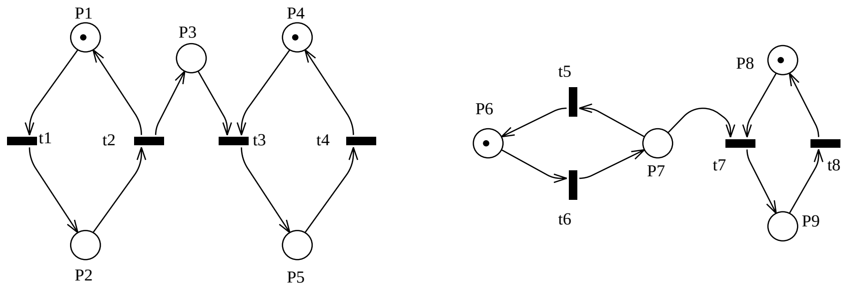


Fig. 8. Examples of two nets that are $\overline{B}LR$ and $B\overline{LR}$, respectively.

Mutual exclusion — Two mutual exclusion properties are of interest: one among places and one among transitions. Two places p and q are mutually exclusive in a Petri net if their token counts cannot be both positive in the same marking, i.e., $\forall \boldsymbol{m} \in RS \;\; \boldsymbol{m}(p) \cdot \boldsymbol{m}(q) = 0$. Two transitions in a PN are mutually exclusive if they cannot be both enabled in any marking.

Analysis Techniques - Depending on the techniques used for deriving these properties, they can be classified in the following way:

– *Structural* properties of Petri nets are obtained from the incidence matrix and from the graph structure of the model, independently of the initial marking.

– *Behavioural* properties of Petri nets depend on the initial marking and are obtained from the reachability graph (finite case) of the net or from the coverability tree (infinite case)[5].

Structural properties are quite interesting since they are proved directly from the structure of the model and are thus valid for every possible initial marking.

Linear Algebraic Techniques - Linear algebraic techniques, derive some basic properties of the net from the incidence matrix $\boldsymbol{C}$.

The relevance of the incidence matrix is due to the fact that it allows the net dynamics to be expressed by means of linear algebraic equations. In particular, we can observe that *for any marking* $\boldsymbol{m}$, the firing of a transition t enabled in $\boldsymbol{m}$ produces the new marking

$$\boldsymbol{m}' = \boldsymbol{m} + \boldsymbol{c}(.,t)^{\mathrm{T}} \tag{2}$$

where $\boldsymbol{m}$ and $\boldsymbol{m}'$ are row vectors, and $\boldsymbol{c}(.,t)$ is the column vector of $\boldsymbol{C}$ corresponding to transition t.

A similar relation holds for transition sequences. Given a transition sequence $\sigma = t_{(1)}, \cdots, t_{(k)}$, we define the transition count vector $\boldsymbol{v}_\sigma$ whose $i - th$ entry indicates how many times transition t_i appears in the sequence σ. $\boldsymbol{v}_\sigma$ is a $|T|$-component column vector. The marking $\boldsymbol{m}^{"}$ obtained by firing the transition sequence σ from marking $\boldsymbol{m}$ ($\boldsymbol{m}[\sigma\rangle\boldsymbol{m}^{"}$) can be obtained using the following equation:

$$\boldsymbol{m}^{"} = \boldsymbol{m} + [\boldsymbol{C}\boldsymbol{v}_\sigma]^{\mathrm{T}} \tag{3}$$

Observe that only the number of times a transition fires is important: the order in which transitions appear in σ is irrelevant. The order is important for the definition of the transition sequence, and for checking whether the sequence can be fired, but it plays no role in the computation of the marking reached by that sequence. This remark leads to important consequences related to the definition of invariant relations for PN models.

P-semiflows and P-invariant relations — A Petri net is *strictly conservative* (or strictly invariant) iff

$$\sum_{p=1}^{P} m_p = \sum_{p=1}^{P} m_{0p}, \qquad \forall \boldsymbol{m} \in RS(\boldsymbol{m}_0)$$

Let us define a $|P|$-component weight column vector $\boldsymbol{y} = [y_1, y_2, \cdots, y_{|P|}]^{\mathrm{T}}$, whose entries are natural numbers. Consider the scalar product between the row

[5] The coverability tree provides a finite representation for infinite reachability graphs based on partial information. Details on this structure and on the algorithms for its construction can be found in [59].

vector representing an arbitrary marking $\boldsymbol{m}'$, and $\boldsymbol{y}$ (denoted $\boldsymbol{m}' \cdot \boldsymbol{y}$). A Petri net is *conservative* (or P invariant) iff

$$\exists\, \boldsymbol{y} \;=\; (y_1,\; y_2,\; ...,\; y_P) > 0 \textit{ such that}$$

$$\sum_{p=1}^{P} y_p m_p \;=\; \sum_{p=1}^{P} y_p m_{0p} \qquad \forall \boldsymbol{m} \in RS(\boldsymbol{m}_0)$$

If $\boldsymbol{m}[t\rangle\boldsymbol{m}'$, then using (2) we can rewrite $\boldsymbol{m}' \cdot \boldsymbol{y}$ as:

$$\boldsymbol{m}' \cdot \boldsymbol{y} = \boldsymbol{m} \cdot \boldsymbol{y} + \boldsymbol{c}(.,t)^{\mathrm{T}} \cdot \boldsymbol{y} \tag{4}$$

Obviously, if $\boldsymbol{c}(.,t)^{\mathrm{T}} \cdot \boldsymbol{y} = 0$, the weighted token count in the Petri net (using the entries of $\boldsymbol{y}$ as weights) is the same for $\boldsymbol{m}$ and $\boldsymbol{m}'$. This means that the weighted token count is *invariant* with respect to the firing of t. More generally, if

$$\boldsymbol{C}^{\mathrm{T}} \cdot \boldsymbol{y} = \boldsymbol{0} \tag{5}$$

i.e., vector $\boldsymbol{y}$ is an integer solution of the set of linear equations

$$\forall t \in T: \quad \boldsymbol{c}(.,t)^{\mathrm{T}} \cdot \boldsymbol{y} = \boldsymbol{0} \tag{6}$$

then, no matter what sequence of transitions fires, the weighted token count does not change, and remains the same for any marking reachable from any given initial marking $\boldsymbol{m}$. The positive vectors $\boldsymbol{y}$ that satisfy Equation (5) are called the *P-semiflows* of the Petri net. Note that P-semiflows are computed from the incidence matrix, and are thus independent of any notion of initial marking. Markings are only instrumental for the interpretation of P-semiflows.

If $\boldsymbol{y}$ is an arbitrary vector of natural numbers, it can be visualized as a bag of places in which p_i appears with multiplicity y_i. This leads to the expression

$$\forall t \in T: \quad \sum_{p_i \in P} C(p_i, t) \cdot y_i = 0 \tag{7}$$

which identifies an invariant relation, stating that the sum of tokens in all places, weighted by $\boldsymbol{y}$, is constant for any reachable marking, and equal to $\boldsymbol{m}_0 \cdot \boldsymbol{y}$, for any choice of the initial marking $\boldsymbol{m}_0$. This invariant relation is called a place invariant, or simply *P-invariant.*

As a consequence, if in a PN model all places are covered by P-semiflows[6], then for any reachable marking (and independently of the initial marking), the maximum number of tokens in any place is finite (since the initial marking is finite) and the net is said to be *structurally bounded.*

All P-semiflows of a PN can be obtained as linear combinations of the P-semiflows that are elements of a minimal set PS. See [45,49,9,10] for P-semiflows computation algorithms.

[6] A place p is covered by a P-semiflow if there is at least one vector $\boldsymbol{y}$ with a non null entry for p.

T-semiflows and T-invariant relations — As observed in Equation (3), if $\boldsymbol{v}_\sigma$ is a firing count vector of a transition sequence σ, then

$$\boldsymbol{m}' = \boldsymbol{m} + [\boldsymbol{C}\boldsymbol{v}_\sigma]^{\mathrm{T}} \tag{8}$$

Obviously, if $[\boldsymbol{C}\boldsymbol{v}_\sigma]^{\mathrm{T}} = 0$, we obtain that $\boldsymbol{m}' = \boldsymbol{m}$ and we can observe that the firing sequence σ brings the PN back to the same marking $\boldsymbol{m}$. The vectors $\boldsymbol{x}$, that are integer solutions of the matrix equation

$$\boldsymbol{C} \cdot \boldsymbol{x} = \boldsymbol{0} \tag{9}$$

are called T-semiflows of the net. This matrix equation is equivalent to the set of linear equations

$$\forall p \in P: \quad \boldsymbol{c}(p,.) \cdot \boldsymbol{x} = 0 \tag{10}$$

In general, the invariant relation (called transition invariant or *T-invariant*) produced by a T-semiflow is the following:

$$\forall p \in P: \quad \sum_{t \in T} C(p,t) \cdot x(t) = 0 \tag{11}$$

This invariant relation states that, by firing from marking $\boldsymbol{m}$ any transition sequence σ whose transition count vector is a T-semiflow, the marking obtained at the end of the transition sequence is equal to the starting one, provided that σ can actually be fired from marking $\boldsymbol{m}$ ($\boldsymbol{m}[\sigma\rangle\boldsymbol{m}$). A net covered by T semiflows may have home states. A net with home states is covered by T-semiflows.

Note again that the T-semiflows computation is independent of any notion of marking, so that T-semiflows are identical for all PN models with the same structure and different initial markings.

Observe the intrinsic difference between P- and T-semiflows. The fact that all places in a Petri net are covered by P-semiflows is a sufficient condition for boundedness, whereas the existence of T-semiflows is only a necessary condition for a PN model to be able to return to a starting state, because there is no guarantee that a transition sequence with transition count vector equal to the T-semiflow can actually be fired.

Like P-semiflows, all T-semiflows can be obtained as linear combinations of the elements of a minimal set TS.

5 Petri Nets with Priority

A marked Petri net with transition priorities, arc multiplicities, and inhibitor arcs can be formally defined by the following tuple:

$$PN \;=\; (P,\, T,\, \Pi(.),\, I(.),\, O(.),\, H(.),\, \boldsymbol{m}_0)$$

– P is a set of places,
– T is a set of transitions,
– $\boldsymbol{m}_0$ is an initial marking,
– $\Pi(.)$, $I(.)$, $O(.)$, $H(.)$ are four functions defined on T.

The priority function $\Pi(.)$ maps transitions into non-negative natural numbers representing their priority level. The input, output, and inhibition functions $I(.)$, $O(.)$, and $H(.)$ map transitions on "bags" of places. The former two are represented as directed arcs from places to transitions and viceversa; the inhibition function is represented by circle-headed arcs. When greater than one, the multiplicity is written as a number next to the corresponding arc.

The priority definition that we assume in this paper is global: the enabled transitions with a given priority k always fire before any other enabled transition with priority[7] $j < k$.

This kind of priority definition can be used for two different modelling purposes: (1) it allows the partition of the transition set into classes representing actions at different logical levels, e.g. actions that take time versus actions corresponding to logical choices that occur instantaneously; (2) it gives the possibility of specifying a deterministic conflict resolution criterion.

Enabling and firing — The firing rule in Petri nets with priority requires the following new definitions:

– a transition t_j is said to *have concession* in marking $\boldsymbol{m}$ if the numbers of tokens in its input and inhibitor places verify the usual enabling conditions for PN models without priority $(\boldsymbol{m} \geq I(t)) \bigwedge (\boldsymbol{m} < H(t))$;
– a transition t_j is said to *be enabled* in marking $\boldsymbol{m}$ if it has concession in the same marking, and if no transition $t_k \in T$ of priority $\pi_k > \pi_j$ exists that has concession in $\boldsymbol{m}$. As a consequence, two transitions may be simultaneously enabled in a given marking only if they have the same priority level;
– a transition t_j can *fire* only if it is enabled. The effect of transition firing is identical to the case of PN models without priority.

Note that the presence of priority only restricts the set of enabled transitions (and therefore the possibilities of firing) with respect to the same PN model without priority. This implies that some properties are not influenced by the addition of a priority structure, while others are changed in a well-determined way, as we shall see in a while.

5.1 Conflicts, Confusion, and Priority

The notions of conflict and confusion are modified when a priority structure is associated with transitions. It is thus very important to be able to clearly identify by inspection of the net structure the sets of potentially conflicting transitions.

[7] Without loss of generality, we also assume that all lower priority levels are not empty, i.e.:

$$\forall t_j \in T, \quad \pi_j > 0 \implies \exists t_k \in T \ : \ \pi_k = \pi_j - 1$$

Conflict — The notion of conflict is drastically influenced by the introduction of a priority structure in PN models. The definition of effective conflict has to be modified with respect to the new notion of concession. Instead, the definition of enabling degree given in Section 4 remains unchanged for PN models with priority. Observe that this implies that both, transitions that have concession and enabled transitions, have enabling degree greater than zero. Conflict resolution causes the enabling degree of some transition to be reduced, and this may happen both for transitions with concession and for enabled ones. Hence the definition of the effective conflict relation is modified as follows.

Definition 9. *Transition t_i is in* effective conflict *relation with transition t_j in marking $\boldsymbol{m}$, (t_i $EC(\boldsymbol{m})$ t_j) iff t_j has concession in $\boldsymbol{m}$, t_i is enabled in $\boldsymbol{m}$, and the enabling degree of t_j decreases after the firing of t_i.*

Observe that a necessary condition for the EC relation to hold is that $\pi_i \geq \pi_j$, otherwise t_i would not be enabled in $\boldsymbol{m}$.

The definition of different priority levels for transitions introduces a further complication, since it destroys the locality of conflicts typical of PN models without priority. This observation leads to the possibility of *indirect conflicts*.

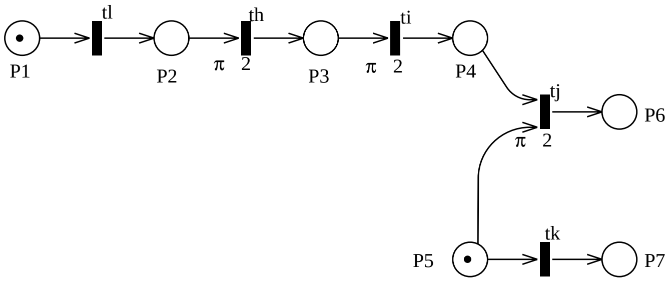


Fig. 9. An example of indirect conflict

Let us consider the net in Fig. 9. Transitions t_l and t_k are both enabled in the marking represented in the figure (since they both have concession, and no higher priority transition has concession), and apparently they are not in conflict, since they do not share input or inhibition places. According to the definition of concurrent transitions given in Section 4.1, one might conclude that t_l and t_k are concurrent. However, the firing of t_l enables a sequence of transitions t_h, t_i, and t_j which have higher priority than t_k, so that:

1. transition t_k becomes disabled while keeping its concession;
2. transition t_h is certainly the next transition to fire;
3. the firing of t_j removes the token from place p_5, thus taking concession away from transition t_k.

This sequence of events is not interruptible after the firing of t_l, due to the priority structure, and eventually results in the disabling of t_k through the firing of higher priority transitions. We call this situation *indirect effective conflict* between t_l and t_k.

Definition 10. *For any priority PN model $\mathcal{M}_\pi$, $\forall t_i, t_j \in T$ such that $t_i \neq t_j$, $\forall \boldsymbol{m} : P \to \mathbb{N}$, transition t_i is in indirect effective conflict with t_j in marking $\boldsymbol{m}$ (denoted $t_i\ IEC(\boldsymbol{m})\ t_j$) iff*

- *t_j has concession in $\boldsymbol{m}$*
- *$t_i \in E(\boldsymbol{m})$*
- *$\exists \sigma = t_{(1)}, \ldots, t_{(k)}$ such that*
  1. *$\boldsymbol{m}[t_i\rangle \boldsymbol{m}_{(1)}[t_{(1)}\rangle \ldots \boldsymbol{m}_{(k)}[t_{(k)}\rangle \boldsymbol{m}'$, and*
  2. *$\forall 1 \leq h \leq k, \pi_{(h)} > \pi_j$, and*
  3. *$t_{(k)} EC(\boldsymbol{m}_{(k)}) t_j$.*

Confusion and priority — In Section 4.1, the concept of confusion was discussed in the framework of PN models without priority. In this section we shall see how the introduction of a priority structure can avoid confusion.

Confusion is an important notion because it highlights the fact that, in terms of event ordering, the system behaviour is not completely defined: this under-specification could be due either to a precise modelling choice (the chosen abstraction level does not include any information on the ordering of events, hence the model analysis must investigate the effect of pursuing any possible ordering) or to a modelling error. The introduction of a priority structure may force a deterministic ordering of conflict resolutions that removes confusion.

For instance, let us consider again the example depicted in Fig. 9 assuming first that transitions t_l and t_k have the same priority level of the others. A confusion situation arises due to the fact that both sequences $\sigma = t_k, t_l, t_h, t_j$ and $\sigma' = t_l, t_h, t_i, t_k$ are fireable in marking $\boldsymbol{m} = p_1 + p_5$, and that they involve different conflict resolutions. By making the priority level of transitions t_h, t_i, and t_j higher than that of t_l and t_k we have removed the confusion situation since any conflict between t_j and t_k is always solved in favour of t_j; as a consequence the sequence $\sigma' = t_l, t_h, t_i, t_k$ is not fireable in $\boldsymbol{m}$ and confusion is avoided.

A structural necessary condition for indirect conflict is the presence of a non free-choice conflict comprising at least two transitions t_i and t_j at the same priority level and a third transition t_k causally connected to either t_i or t_j and such that $\pi_k = \pi_i = \pi_j$.

Structural conflict — For PN models without priority, we defined the notion of *structural conflict* relation (SC) to identify potentially conflicting pairs of transitions by inspection of the net structure. Intuitively, two transitions t_i and t_j are in structural conflict if they share at least one input place or if the output set of t_i is not disjoint from the inhibition set of t_j. This definition does not change for Petri nets with priority.

In this case we can also define the *indirect structural conflict* (ISC) relation that gives a necessary condition for two transitions to be in indirect effective conflict relation in some marking. Intuitively, we have first to find a pair of transitions t_j and t_k such that $\pi_j > \pi_k$ and $t_j SC t_k$; then we have to follow the net arcs backwards starting from transition t_j until we find a new transition t_l such that $\pi_l \leq \pi_k$. All the transitions on the path (including t_l) with priority greater than or equal to π_k, are in indirect structural conflict relation with t_k. Indeed, any transition on the path can trigger a firing sequence of transitions with priority higher than π_k, that may eventually enable transition t_j whose firing would decrease the enabling degree of transition t_k. Notice that the transitions on the path are in causal connection relation. A formal recursive definition for the ISC relation follows:

Definition 11. *Given a priority PN model, two transitions t_l and t_k are in indirect structural conflict relation (denoted t_l ISC t_k) iff*

- $\pi_l \geq \pi_k$;
- $\exists t_j : \, (\pi_j > \pi_k) \wedge (t_k SCC t_j) \wedge ((t_j SC t_k) \vee (t_j ISC t_k))$

where SCC is the structural causal connection relation defined in the previous section.[8]

Let us consider the model of Fig. 9 again; since transition t_j is in SC relation with transition t_k, $\pi_j > \pi_k$ and $t_l SCC t_j$, it is possible to conclude that $t_l ISC t_k$.

Given the definition of structural conflict, it is possible to introduce the notion of *conflict set* and *extended conflict set* whose motivations will become apparent in the following sections.

We define the *symmetric structural conflict* as follows:

Definition 12. *Transition t_i is in* symmetric structural conflict *with t_j (denoted t_i SSC t_j) iff*

- $\pi_i = \pi_j$ *and*
- $t_i SC t_j \;\vee\; t_j SC t_i \;\vee\; t_i ISC t_j \;\vee\; t_j ISC t_i.$

The conflict set associated with a given transition t_i is the set of transitions that might be in conflict with t_i in some marking.

Definition 13. *The conflict set associated with a given transition t_i is defined as*

$$CS(t_i) = \{t_j : (t_i SSC t_j)\}$$

The transitive closure of the *SSC* relation is an equivalence relation that allows the partition of the set T into equivalence classes called *extended conflict sets.*

Definition 14. $ECS(t_i) = \{t_j : t_i \; SSC^* \; t_j \wedge \sim (t_i \; SME \; t_j)\}.$

[8] If the PN allows the presence of inhibitor arcs, the *SCC* relation must also account for the fact that t_k is causally connected with t_j also in case its firing decreases the marking of some of the places that are part of the (non-empty) inhibitor set of t_j.

In any marking that enables transitions of the same *ECS*, a choice that may have effect on the future evolution of the net must be made in order to decide which, among these transitions, has to be fired next

Two simultaneously enabled transitions t_i and t_j that belong to different *ECS* can be fired in any order.

5.2 Properties of Petri Nets with Priority

In order to briefly discuss the impact that priorities have on the properties of PN models, we must first divide them into two broad classes. Properties that hold for all states in the state space are called *safety* or *invariant* properties; properties that instead hold only for some state in the state space are called *eventuality* or *progress* properties). Examples of invariant properties are boundedness, and mutual exclusion. Examples of eventuality properties are reachability (a given marking will be eventually reached) and liveness (a transition will eventually become enabled).

Let $\mathcal{M}_\pi$ be a PN model with priority and let $\mathcal{M}$ be the underlying PN model without priority. Since the introduction of priority can only reduce the state space, all the safety properties that can be shown to be true for $\mathcal{M}$, surely hold also for $\mathcal{M}_\pi$. Eventuality properties instead are not preserved in general by the introduction of a priority structure.

It is interesting to observe that P and T-invariants describe properties that continue to hold after the addition of a priority structure. The reason is that they are computed only by taking into account the state change caused by transition firing, without any assumption on the possibility for a transition of ever becoming enabled. Boundedness is preserved by the introduction of a priority structure in the sense that a bounded PN model remains bounded after the introduction of a priority specification. This implies that the use of P-semiflows to study the boundedness of a PN model can be applied to the model without priority $\mathcal{M}$ associated with a priority PN model $\mathcal{M}_\pi$ and if the former model is shown to be structurally bounded, the conclusion can be extended to the latter model. Observe, however, that an unbounded PN model may become bounded after the specification of an appropriate priority structure.

On the other hand, since enabling is more restricted than in the corresponding PN model without priority, reachability is not preserved in general by the addition of a priority structure. However, a marking $\boldsymbol{m}'$ is reachable from a marking $\boldsymbol{m}$ in a PN model with priority only if it is reachable in the corresponding PN model without priority.

Liveness is intimately related to the enabling and firing rules, hence it is greatly influenced by a change in the priority specification: a live PN model may become not live after the introduction of an inappropriate priority structure and, viceversa, a PN model that is not live, may become live after the addition of an appropriate priority structure.

Expressive Power - According to their definition, P/T nets do not allow modelling zero tests, i.e., transitions that are enabled only if some place is empty.

The introduction of priorities and inhibitor arcs yields such a feature, making the extended model less amenable for analysis as we have just seen during the discussion of the properties of Petri nets with priorities. On the other hand, with the addition of inhibitor arcs and priorities, Petri nets increase their modelling power, actually leading to Turing machines [59].

It is possible to implement a system with inhibitor arcs using priorities and viceversa, even in the unbounded case, while preserving concurrent semantics, so both extensions can be interchanged except for modelling convenience (the transformations are rather cumbersome [23]). Inhibitor arcs have the advantage that they are graphically represented in the net structure, while the influence of a priority definition on the enabling of some transition is not so clearly reflected and is not so local. On the other hand, priorities arise naturally when a timing interpretation is considered. Therefore, despite their formal equivalence, both extensions are allowed on equal footing, because they have been introduced to cope with different situations. Figure 10 shows an example in which the repeated firing of t_1 accumulates tokens in P_1 until the operation mode switches and all the accumulated tokens are consumed (repeated firing of t_2). When P_1 becomes empty t_3 fires and the system goes back in the initial state. The representations with inhibitor arcs and priorities yield exactly the same behaviour.

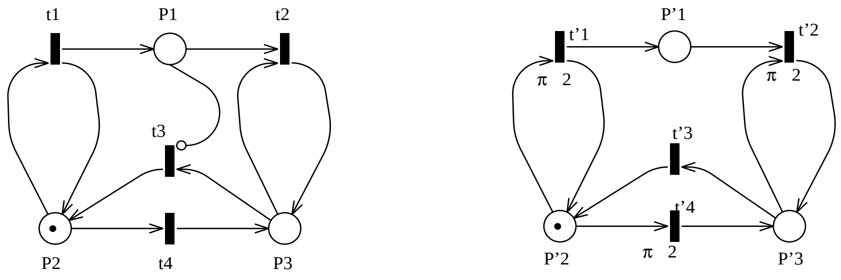


Fig. 10. Equivalence between inhibitor arcs and priority specifications

Inhibitor arcs deriving from bounded places can be implemented with the use of multiplicity and complementary places (preserving the interleaving semantics), as shown in the example of Fig. 11, which is equivalent to that of Fig. 10 assuming that place P_1 cannot contain more than 10 tokens. This simple transformation does not work so well, however, when concurrent semantics is considered (see [12] for details).

6 Time in Petri Nets

In this section we discuss the issues related to the introduction of *temporal concepts* into PN models. Particular attention will be given to the temporal seman-

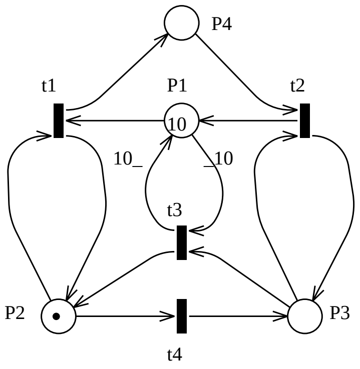


Fig. 11. Representations of inhibitor arcs with complementary places

tics that is peculiar to stochastic PNs (SPNs) and generalized SPNs (GSPNs). For this reason we shall always assume that timed transitions are associated with temporal specifications such that the simultaneous firing of two or more timed transitions can be neglected (this event has probability zero in SPNs and GSPNs).

6.1 The Motivations for Timing

The PN models that were considered in the previous sections included no notion of time. The concept of time was intentionally avoided in the original work by C.A.Petri [60], because of the effect that timing may have on the behaviour of PNs. In fact, the association of timing constraints with the activities represented in PN models may prevent certain transitions from firing, thus destroying the important assumption that all possible behaviours of a real system are represented by the structure of the PN.

In [59], the first book on PNs that dealt extensively with applications, the only remark about timed PNs was the following: "*The addition of timing information might provide a powerful new feature for PNs, but may not be possible in a manner consistent with the basic philosophy of PNs*". This attitude towards timing in PN models is due to the fact that PNs were originally considered as formal automata and investigated in their theoretical properties. Most of the early questions raised by researchers thus looked into the fundamental properties of PN models, into their analysis techniques and the associated computational complexity, and into the equivalence between PNs and other models of parallel computation. When dealing with these problems, timing is indeed not relevant.

Very soon PNs were however recognized as possible models of real concurrent systems, capable of coping with all aspects of parallelism and conflict in asynchronous activities with multiple actors. In this case, timing is not important when considering only the logical relationships between the entities that are part of the real system. The concept of time becomes instead of paramount importance when the interest is driven by real applications whose efficiency is always a relevant design problem. Indeed, in areas like hardware and computer architecture design, communication protocols, and software system analysis, timing is crucial even to define the logical aspects of the dynamic operations.

Time is introduced in Petri nets to model the interaction among several activities considering their starting and completion instants The introduction of time specifications corresponds to an interpretation of the model by means of the observation of the autonomous (untimed) model and the definition of a non-autonomous model.

The pioneering works in the area of timed PNs were performed by P.M.Merlin and D.J.Farber [50], and by J.D.Noe and G.J.Nutt [58]. In both cases, PNs were not viewed as a formalism to statically model the logical relationships among the various entities that form a real system, but as a tool for the description of the global behaviour of complex structures. PNs were used to *tell* all the possible stories that the system can experience, and the temporal specifications were an essential part of the picture.

When introducing time into PN models, it would be extremely useful not to modify the basic behaviour of the underlying untimed model. By so doing, it is possible to study the timed PNs exploiting the properties of the basic model as well as the available theoretical results. The addition of temporal specifications therefore must not modify the unique and original way of expressing synchronization and parallelism that is peculiar to PNs. This requirement obviously conflicts with the user's wishes for extensions of the basic PN formalism to allow a direct and easy representation of specific phenomena of interest. Time specifications are also used to provide ways of reducing the non-determinism of the model by means of rules based on time considerations. Finally, time extensions must provide methods for the computation of performance indices.

Different ways of incorporating timing information into PN models have been proposed by many researchers during the last two decades; the different proposals are strongly influenced by the specific application fields and can be summarized as follows:

- **Timed places** - time may be associated with *places*:
  - tokens generated in an output place become available to fire a transition only after a delay has elapsed; the delay is an attribute of the place.
- **Timed tokens**- time may be associated with *tokens*:
  - tokens carry a time-stamp that indicates when they are available to fire a transition; this time-stamp can be incremented at each transition firing.
- **Timed arcs** - time may be associated with *arcs*:
  - a travelling delay is associated with each arc; tokens are available for firing only when they reach a transition.
- **Timed transitions** - time may be associated with *transitions*:
  - activity start corresponds to transition enabling,
  - activity end corresponds to transition firing.

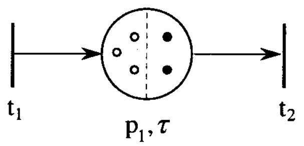


Fig. 12. Timed places

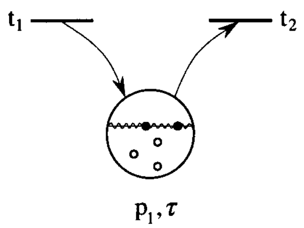


Fig. 13. Timed tokens

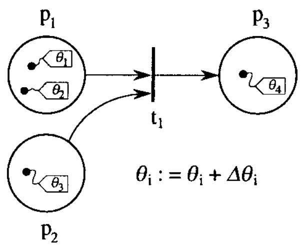


Fig. 14. Timed arcs

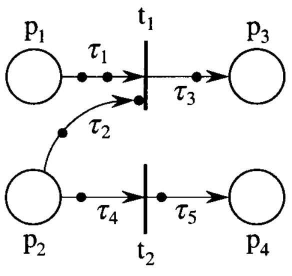


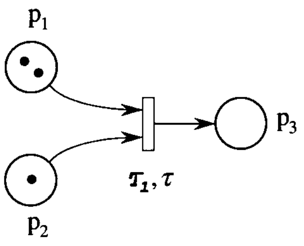
Fig. 15. Timed transitions

Fig.s 12, 13, 14, and 15, depicts in a compact manner these different possibilities with the purpose of helping the reader in grasping the differences between the various extensions that are sometimes subtle and difficult to identify at first glance. The reader interested in understanding better the implications of the different proposals is referred to the original papers [62,71,44,61,50,65,32,70].

6.2 Timed Transitions

Timed transitions represent the most common extension used by the authors to add time to PN models. The firing of a transition in a PN model corresponds to the event that changes the state of the real system. This change of state can be due to one of two reasons: it may either result from the verification of some logical condition in the system, or be induced by the completion of some activity. Considering the second case, we note that transitions can be used to model activities, so that transition enabling periods correspond to activity executions and transition firings correspond to activity completions. Hence, time can be naturally associated with transitions.

Different firing policies may be assumed: the *three-phase firing* assumes that tokens are consumed from input places when the transition is enabled, then the delay elapses, finally tokens are generated in output places; *atomic firing* assumes instead that tokens remain in input places during the whole transition delay; they are consumed from input places and generated in output places when the transition fires.



Timed transition Petri nets (TTPN) with atomic firing can preserve the basic behaviour of the underlying untimed model. It is thus possible to qualitatively study TTPN with atomic firing exploiting the theory developed for untimed (autonomous) PN (reachability set, invariants, etc.). Timing specifications may affect the qualitative behaviour of the PN only when they describe *constant* and *interval* firing delays.

We can explain the behaviour of a timed transition (whose graphical representation is usually a box or a thick bar and whose name usually starts with T) by assuming that it incorporates a timer. When the transition is enabled, its local clock is set to an initial value. The timer is then decremented at constant speed, and the transition fires when the timer reaches the value zero. The timer associated with the transition can thus be used to model the duration of an activity whose completion induces the state change that is represented by the change of marking produced by the firing of T. The type of activity associated with the transition, whose duration is measured by the timer, depends on the DEDS that we are modelling: it may correspond to the execution of a task by a processor, or to the transmission of a message in a communication network, or to the work performed on a part by a machine tool in a manufacturing system. It is important to note that the activity is assumed to be in progress while the transition is enabled. This means that in the evolution of more complex nets, an interruption of the activity may take place if the transition loses its enabling condition before it can actually fire. The activity may be resumed later on, during the evolution of the net in the case of a new enabling of the associated transition. This may happen several times until the timer goes down to zero and the transition finally fires.

It is possible to define a *timed transition sequence* or *timed execution* of a timed PN system as a transition sequence (as defined in Section 4.1) augmented with a set of nondecreasing real values describing the epochs of firing of each transition. Such a timed transition sequence is denoted as follows:

$$[(\tau_{(1)}, T_{(1)}); \cdots ; (\tau_{(j)}, T_{(j)}); \cdots]$$

The time intervals $[\tau_{(i)}, \tau_{(i+1)})$ between consecutive epochs represent the periods during which the PN sojourns in marking $\boldsymbol{m}_{(i)}$. This sojourn time corresponds to a period in which the execution of one or more activities is in progress and the state of the system does not change.

6.3 Immediate Transitions

As we noted before, not all the events that occur in a DEDS model correspond to the end of time-consuming activities (or to activities that are considered time-consuming at the level of detail at which the model is developed). For instance, a model of a multiprocessor system described at a high level of abstraction often neglects the durations of task switchings, since these operations require a very small amount of time, compared with the durations of task executions. The same can be true for bus arbitration compared with bus use. In other cases, the state change induced by a specific event may be quite complex, and thus difficult to obtain with the firing of a single transition. Moreover, the state change can depend on the present state in a complex manner. As a result, the correct evolution of the timed PN model can often be conveniently described with subnets of transitions that consume no time and describe the logics or the algorithm of state evolution induced by the complex event.

To cope with both these situations in timed PN models, it is convenient to introduce a second type of transition called *immediate*. Immediate transitions fire as soon as they become enabled (with a null delay), thus acquiring a sort of precedence over timed transitions. In this paper, immediate transitions are depicted as thin bars whereas timed transitions are depicted as boxes or thick bars.

6.4 Parallelism and Conflict

The introduction of temporal specifications in PN models must not reduce the modelling capabilities with respect to the untimed case. Let us verify this condition as far as parallelism and conflict resolution are considered.

Pure parallelism can be modelled by two transitions that are independently enabled in the same marking. The evolution of the two activities is measured by the decrement of the clocks associated with the two transitions. When one of the timers reaches zero, the transition fires and a new marking is produced. In the new marking, the other transition is still enabled and its timer can either be reset or not depending on the different ways of managing this timer that will be discussed in the next section.

Consider now transitions T_1 and T_2 in Fig. 16. In this case, the two transitions are in free-choice conflict. In untimed PN systems, the choice of which of the two transitions to fire is completely nondeterministic. When more than one timed transition with atomic firing is enabled, the behaviour is similar, but a problem arises: *Which one of the enabled transitions is going to fire?* Two alternative selection rules are possible:

- **preselection** - the enabled transition that will fire is chosen when the marking is entered, according to some metric (e.g., priority),

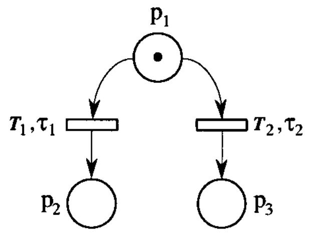


Fig. 16. Conflicting transitions

– **race-** the enabled transition that will fire is the one whose firing delay is minimum.

In the case of timed PNs, the conflict resolution depends on the delays associated with transitions and is obtained through the so-called *race* policy: when several timed transitions are enabled in a given marking $\boldsymbol{m}$, the transition with the shortest associated delay fires first (thus disabling the possible conflicting transitions).

Having discussed the conflict resolution policy among timed transitions, it is important to emphasize the fact that, when two or more immediate transitions are enabled in the same marking, some rule must be specified to select the one to fire first, thus ordering the firing of immediate transitions. Two types of rules will be used when situations of this type occur in the following. The first one is based on a deterministic choice of the transition to fire using the mechanism of *priority*. A second mechanism consists in the association of a discrete probability distribution function with the set of conflicting transitions. In this case the conflicts among immediate transitions are randomly solved.

In some cases, however, it may be desirable to separate conflict resolution from timing specification of transitions. Immediate transitions can be used to obtain this separation. The conflict can be transferred to a barrier of conflicting immediate transitions, followed by a set of timed transitions. The extensive use of this technique can eliminate from a net all conflicts among timed transitions that are simultaneously enabled in a given marking. If this mechanism is consistently used to prevent timed transitions from entering into conflict situations, a *preselection* policy of the (timed) transition to fire next is said to be used.

Conflicts comprising timed and immediate transitions have an important use in timed PNs: they allow the interruption (or preemption) of ongoing activities, when some special situation occurs. Consider, for example, the subnet in Fig. 17. A token in place p_1 starts the activity modelled by timed transition T_1. If a token arrives in p_2 before the firing of T_1, immediate transition t_2 becomes enabled and fires, thus disabling timed transition T_1. This behaviour again provides an example of the precedence of immediate over timed transitions.

The presence of immediate transitions induces a distinction among markings. Markings in which no immediate transitions are enabled are called *tangible*, whereas markings enabling at least one immediate transition are said to be

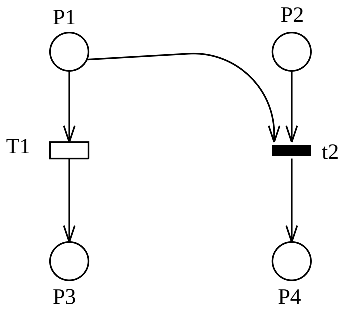


Fig. 17. Interrupting an activity with an immediate transition

vanishing. The timed PN system spends a positive amount of time in tangible markings, and a null time in vanishing markings.

6.5 Memory

An important issue that arises at every transition firing, when timed transitions are used in a model, is how to manage the timers of all the transitions that do not fire.

From the modeling point of view, the different policies that can be adopted link the past history of the systems to its future evolution considering various ways of retaining *memory* of the time already spent in activities. The question concerns the *memory policy* of transitions, and defines how to set the transition timers when a state change occurs, possibly modifying the enabling of transitions. Two basic mechanisms can be considered for a timed transition at each state change.

- **Continue**. The timer associated with the transition holds the present value and will *continue* later on the count-down.
- **Restart**. The timer associated with the transition is *restarted*, i.e., its present value is discarded and a new value will be generated when needed.

To model the different behaviours arising in real systems, different ways of keeping track of the past are possible by associating different continue or restart mechanisms with timed transitions. We discuss here three alternatives:

- **Resampling**. At each and every transition firing, the timers of all the timed transitions are discarded (restart mechanism). No memory of the past is recorded. After discarding all the timers, new values of the timers are set for the transitions that are enabled in the new marking.
- **Enabling memory**. At each transition firing, the timers of all the timed transitions that are disabled are restarted whereas the timers of all the timed

transitions that are not disabled hold their present value (continue mechanism). The memory of the past is recorded with an *enabling memory variable* associated with each transition. The enabling memory variable accounts for the work performed by the activity associated with the transition since the last instant of time its timer was set. In other words, the enabling memory variable measures the enabling time of the transition since the last instant of time it became enabled.

– **Age memory**. At each transition firing, the timers of all the timed transitions hold their present values (continue mechanism). The memory of the past is recorded with an *age memory variable* associated with each timed transition. The age memory variable accounts for the work performed by the activity associated with the transition since the time of its last firing. In other words, the age memory variable measures the *cumulative* enabling time of the transition since the last instant of time when it fired.

The three memory policies can be used in timed PN models for different modelling purposes. In the first case (resampling) the work performed by activities associated with transitions that do not fire is lost. This may be adequate for modelling, for example, competing activities of the type one may find in the case of the parallel execution of hypothesis tests. The process that terminates first is the one that verified the test; those hypotheses whose verification was not completed become useless, and the corresponding computations need not be saved. The practical and explicit use of this policy is very limited, but it must be considered because of its theoretical importance in the case of SPNs and GSPNs.

The other two policies are of greater importance from the application viewpoint. They can coexist within the same timed PN model, because of the different semantics that can be assigned to the different transitions of the model. For a detailed discussion on this topic the reader is referred to [2,31].

6.6 Multiple Enabling

Special attention must be paid to the timing semantics in the case of timed transitions with enabling degree larger than one. Different semantics are possible when several tokens are present in the input places of a transition. Borrowing from queueing network terminology, we can consider the following different situations.

1. **Single-server semantics**: a firing delay is set when the transition is first enabled, and new delays are generated upon transition firing if the transition is still enabled in the new marking.
2. **Infinite-server semantics**: every enabling set of tokens is processed as soon as it forms in the input places of the (timed) transition. Its corresponding firing delay is generated at this time, and the timers associated with all these enabling sets run down to zero in parallel.
3. **Multiple-server semantics**: enabling sets of tokens are processed as soon as they form in the input places of the transition up to a maximum degree

of parallelism (say K). For larger values of the enabling degree, the timers associated with new enabling sets of tokens are set only when the number of concurrently running timers decreases below the value of K.

A simple example will help the reader to understand the three semantics. Consider a timed transition T with enabling degree 3. Assume also that the net starts to operate at time 0 with such an enabling degree. The three enablings are associated with three activities whose durations are 3, 2, and 4 time units, respectively. We describe next the detailed behaviour of the net, considering the three different semantics with reference to Fig. 18 that illustrates the firing epochs.

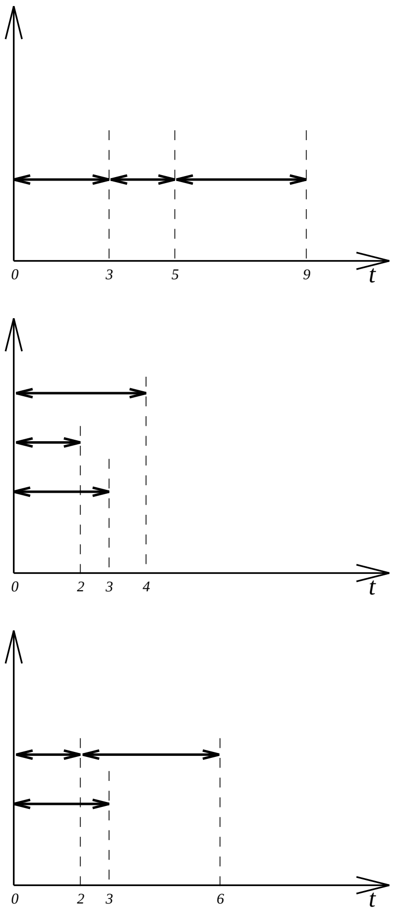


Fig. 18. Firing epochs corresponding to the three different timing semantics

1. Single-server semantics: the serial execution of the activities induces the following sequence of events:
   - t = 0: T_1 is enabled and the first activity starts.
   - t = 3: the first activity (duration 3) ends, T_1 fires and the second activity starts.
   - t = 5: the second activity (duration 2) ends, T_1 fires and the third activity starts.
   - t = 9: the third activity (duration 4) ends and T_1 is disabled.
2. Infinite-server semantics: the parallel execution of the activities induces the following sequence of events:
   - t = 0: T_1 is enabled and all the activities start.
   - t = 2: T_1 fires because of the completion of the second activity (duration 2).
   - t = 3: T_1 fires because of the completion of the first activity (duration 3).
   - t = 4: T_1 fires because of the completion of the third activity (duration 4), and it is disabled.
3. Multiple-server semantics: in this case we assume that the maximum parallelism is $K = 2$. This induces the following sequence of events:
   - t = 0: T_1 is enabled and the first two activities start.
   - t = 2: T_1 fires because of the completion of the second activity (duration 2) thus the third activity can start.
   - t = 3: T_1 fires because of the completion of the first activity (duration 3).
   - t = 6: T_1 fires because of the completion of the third activity (duration 4), and it is disabled.

The introduction of these different firing semantics permits the definition of PN models that are graphically simple without losing any of the characteristics that allow the analysis of their underlying behaviours.

7 Stochastic Petri Nets

Timed Petri nets in which the firing delays are specified by random variables yeld to probabilistic models. The execution of a timed PN model corresponds to a realization of a stochastic point process.

The use of exponential distributions for the definition of temporal specifications is particularly attractive because timed PN in which all the transition delays are exponentially distributed can be mapped onto continuous-time Markov chains (CTMC). In this case the memoryless property of the exponential distribution makes unnecessary the distinction between the distribution of the delay itself, and the distribution of the remaining delay after a change of state.

Stochastic Petri nets are timed (transition) PN with atomic firing and in which transition firing delays are exponentially distributed random variables: each transition t_i is associated with a random firing delay whose probability density function is a negative exponential with rate w_i.

SPNs were originally defined in [37,53] Formally, a SPN model is an 6-tuple

$$\mathrm{SPN} = (P, T, I(.), O(.), W(.), \boldsymbol{m}_0) \tag{12}$$

P, T, $I(.)$, $O(.)$, and $\boldsymbol{m}_0$ have the usual meanings so that the underlying PN model constitutes the structural component of a SPN model.

The function W allows the definition of the stochastic component of a SPN model mapping transitions into real positive functions of the SPN marking. Thus, for any transition t it is necessary to specify a function $W(t, \boldsymbol{m})$. In the case of marking independency, the simpler notation w_k is normally used to indicate $W(t_k)$, for any transition $t_k \in T$. The quantity $W(t_k, \boldsymbol{m})$ (or w_k) is called the "rate" of transition t_k in marking $\boldsymbol{m}$.

In this section we show how SPNs can be converted into Markov chains and how their analysis can be performed to compute interesting performance indices. The construction of the Markov chain associated with a Stochastic Petri Net (SPN) is described first, to set the ground for the subsequent derivation of the probabilistic model associated with a GSPN. Only SPNs with finite state space are considered, as they yield Markov chains that are solvable with standard numerical techniques. More advanced solution methods based on matrix-geometric theory [57] are discussed in [38,57,12].

7.1 The Stochastic Process Associated with a SPN

We have already discussed the motivations behind the work of several authors that led to the proposal of Timed and Stochastic Petri Nets (SPNs). Due to the memoryless property of the exponential distribution of firing delays, it is relatively easy to show [37,53] that SPN systems are isomorphic to continuous time Markov chains (CTMCs). In particular, a k-bounded SPN system can be shown to be isomorphic to a finite CTMC.

Finite State Machine and Marked Graph SPNs - This can be easily seen when the structure of the SPN is that of both a *finite state machine* (no transition has more than one input and one output place) and of a *marked graph* (no place has more than one input and one output transition) with only one token in its initial marking. In this case each place of the net univocally corresponds to a state of the model and the position of the token at a given instant of time identifies the state of the model at that same time. Each place of the net maps into a state of the corresponding CTMC and each transition maps into an arc annotated by the rate of the corresponding firing time distribution. Moreover, if the firing times of the transitions have negative exponential distributions and if the structure of the net is that of a marked graph, the time spent in each place by the net is completely identified by the characteristics of the only transition that may withdraw the token from that place. When the net has the structure of a finite state machine, conflicts among simultaneously enabled transitions arise since several transitions may share the same input place. Since we are assuming that all the activities have negative-exponentially distributed durations, the CTMC corresponding to the SPN is obtained from the net in a straightforward manner. Again each place of the SPN maps into a state of the corresponding CTMC and each transition of the SPN maps into an arc of the CTMC annotated with the rate of the corresponding firing time distribution. Also in this case the time spent by the net in each place has a negative-exponential distribution, but its rate is given by the sum of the firing rates of all the transitions that withdraw tokens from that place.

More complex situations arise, even in this simple case, when several tokens are allowed in the initial marking. These are due to the fact that places no longer correspond to states of the associated probabilistic models. Moreover, in these cases the behaviours of the probabilistic models are also affected by the service selection policies from the input places as well as by the token selection policies adopted at transition firing moments.

Even the very simple case of two tokens waiting in the only input place of a transition raises a set of interesting questions that must be addressed in developing the corresponding probabilistic model. The first has to do with the speed at which the transition withdraws the tokens from its input place in this situation and thus from the service policies discussed in Section 6.6. In the more general case of assuming a form of *load dependency* in which the firing rate of the transition is a function of its enabling degree, an additional specification must be introduced in the model to define the load dependency function associated with each transition. The second question refers to the selection of the token that is removed from the input place upon the firing of the transition. From a "classical" Petri net point of view, this selection policy is inessential since tokens do not carry any identity. In many applications however, it is convenient to associate a physical meaning with the tokens (e.g., customers), so that questions on their flow through the net can be answered. In these situations, when several tokens are simultaneously present in the input place of a transition, if this is assumed to operate with a single server policy, a question on the queueing policy applied to these tokens becomes interesting. The most natural policy (from a Petri net point of view) is a *random order*. When the firing times are exponentially distributed and when the performance figures of interest are only related to the moments of the number of tokens in the input place of a transition, it is possible to show that many queueing policies yield the same results (e.g., random, FIFO, LCFS). It must however be observed that in other cases the choice of the token selection policy may be important and that policies different from the random one must be explicitly implemented through appropriate net constructions.

SPNs with General Structure - In general, the CTMC associated with a given SPN system is obtained by applying the following simple rules:

1. The CTMC state space $S = \{s_i\}$ corresponds to the reachability set $RS(\boldsymbol{m}_0)$ of the PN associated with the SPN ($\boldsymbol{m}_i \leftrightarrow s_i$).

2. The transition rate from state s_i (corresponding to marking $\boldsymbol{m}_i$) to state s_j ($\boldsymbol{m}_j$) is obtained as the sum of the firing rates of the transitions that are enabled in $\boldsymbol{m}_i$ and whose firings generate marking $\boldsymbol{m}_j$.

Based on these simple rules, it is possible to devise algorithms for the automatic construction of the infinitesimal generator (also called the state transition rate matrix) of the isomorphic CTMC, starting from the SPN description.

Assuming that all the transitions of the net operate with a single-server semantics and marking-independent speeds, and denoting with $\mathbf{Q}$ this matrix, with w_k the firing rate of T_k, and with $E_j(\boldsymbol{m}_i) = \{h : T_h \in E(\boldsymbol{m}_i) \wedge \boldsymbol{m}_i[T_h\rangle \boldsymbol{m}_j\}$ the set of transitions whose firings bring the net from marking $\boldsymbol{m}_i$ to marking $\boldsymbol{m}_j$, the components of the infinitesimal generator are:

$$q_{ij} = \begin{cases} \sum_{T_k \in E_j(\boldsymbol{m}_i)} w_k & i \neq j \\ \\ -q_i & i = j \end{cases} \tag{13}$$

where

$$q_i = \sum_{T_k \in E(\boldsymbol{m}_i)} w_k \tag{14}$$

Let $\pi(\boldsymbol{m}_i, \tau)$ be the probability that the SPN is in marking $\boldsymbol{m}_i$ at time τ. The Chapman-Kolmogorov equations for the CTMC associated with an SPN are specified by:

$$\frac{d\pi(\boldsymbol{s}_i, \tau)}{d\tau} = \sum_{\boldsymbol{s}_k} \pi(\boldsymbol{s}_k, \tau) q_{kj} \tag{15}$$

In matrix notation this becomes

$$\frac{d\boldsymbol{\pi}(\tau)}{d\tau} = \boldsymbol{\pi}(\tau)\boldsymbol{Q}, \tag{16}$$

whose solution can be formally written as

$$\boldsymbol{\pi}(\tau) = \boldsymbol{\pi}(0) e^{\boldsymbol{Q}\tau} \tag{17}$$

where $\boldsymbol{\pi}(0)$ is the probability of the initial distribution (in our case we usually have $\pi_i(0) = 1$ if $\boldsymbol{m}_i = \boldsymbol{m}_0$ and $\pi_i(0) = 0$ otherwise) and $e^{\boldsymbol{Q}\tau}$ is the matrix exponentiation formally defined by

$$e^{\boldsymbol{Q}\tau} = \sum_{k=0}^{\infty} \frac{(\boldsymbol{Q}\tau)^k}{k!} \tag{18}$$

In this section we consider only SPNs originating homogeneous and ergodic CTMC. A k-bounded SPN system is said to be ergodic if it generates an ergodic CTMC; it is possible to show that a SPN system is ergodic if $\boldsymbol{m}_0$, the initial marking, is a home state (see Section 2).

If the SPN is ergodic, the steady-state probability distribution on its markings exists and is defined as the limit $\boldsymbol{\pi} = \lim_{\tau\to\infty} \boldsymbol{\pi}(\tau)$. Its value can be computed solving the usual system of linear equations:

$$\begin{cases} \boldsymbol{\pi}\, \boldsymbol{Q} = \boldsymbol{0} \\ \\ \boldsymbol{\pi}\, \boldsymbol{1}^{\mathrm{T}} = 1 \end{cases} \tag{19}$$

where $\boldsymbol{0}$ is a vector of the same size as $\boldsymbol{\pi}$ and with all its components equal to zero and $\boldsymbol{1}^{\mathrm{T}}$ is a vector (again of the same size as $\boldsymbol{\pi}$) with all its components equal to one, used to enforce the normalization condition.

To keep the notation simple, in the rest of this section we will use $\pi_i(\tau)$ and π_i instead of $\pi(\boldsymbol{m}_i, \tau)$ and $\pi(\boldsymbol{m}_i)$ to denote the transient and steady state probabilities of marking $\boldsymbol{m}_i$. The *sojourn time* is the time spent by the PN in a given marking $\boldsymbol{m}$. As we already observed, the Markovian property of this model ensures that the sojourn time in the $i-th$ marking is exponentially distributed with rate q_i. The pdf of the sojourn time in a marking corresponds to the pdf of the minimum among the firing times of the transitions enabled in the same marking; it thus follows that the probability that a given transition $T_k \in E(\boldsymbol{m}_i)$ fires (first) in marking $\boldsymbol{m}_i$ can be expressed as follows:

$$P\{T_k|\boldsymbol{m}_i\} = \frac{w_k}{q_i}. \tag{20}$$

Using the same argument, we can observe that the average sojourn time in marking $\boldsymbol{m}_i$ is given by the following expression:

$$SJ_i = \frac{1}{q_i}. \tag{21}$$

SPN Performance Indices - The steady-state distribution $\boldsymbol{\pi}$ is the basis for a quantitative evaluation of the behaviour of the SPN that is expressed in terms of performance indices. These results can be computed using a unifying approach in which proper index functions (also called *reward functions*) are defined over the markings of the SPN and an average reward is derived using the steady-state probability distribution of the SPN. Assuming that $r(\boldsymbol{m})$ represents a reward function, the average reward can be computed using the following weighted sum:

$$E[R] = \sum_{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0)} r(\boldsymbol{m}_i)\, \pi_i \tag{22}$$

Different interpretations of the reward function can be used to compute different performance indices. In particular, the following quantities can be easily computed.

(1) The *probability of a particular condition* of the SPN. Assuming that *condition* $\Upsilon(\boldsymbol{m})$ is *true* only in certain markings of the PN, we can define the following reward function:

$$r(\boldsymbol{m}) = \begin{cases} 1 & \Upsilon(\boldsymbol{m}) = true \\ \\ 0 & otherwise \end{cases} \tag{23}$$

The desired probability $P\{\Upsilon\}$ is then computed using Equation (22). The same result can also be expressed as:

$$P\{\Upsilon\} \;=\; \sum_{\boldsymbol{m}_i \in A} \pi_i \tag{24}$$

where $A \;=\; \{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0) : \Upsilon(\boldsymbol{m}_i) = \mathit{true}\}$.

(2) The *expected value of the number of tokens in a given place.* In this case the reward function $r(\boldsymbol{m})$ is simply the value of the marking of that place (say place j):

$$r(\boldsymbol{m}) \;=\; n \;\text{ iff }\; m(p_j) \;=\; n \tag{25}$$

Again this is equivalent to identifying the subset $A(j,n)$ of $RS(\boldsymbol{m}_0)$ for which the number of tokens in place p_j is n ($A(j,n) \;=\; \{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0) : \boldsymbol{m}_i(p_j) \;=\; n\}$); the expected value of the number of tokens in p_j is given by:

$$E[m(p_j)] \;=\; \sum_{n>0} [n \; P\{A(j,n)\}] \tag{26}$$

where the sum is obviously limited to values of $n \leq k$, if the place is k-bounded.

(3) The *mean number of firings per unit of time of a given transition.* Assume that we want to compute the firing frequency of transition T_j (the *throughput* of T_j); observing that a transition may fire only when it is enabled, we have that the reward function assumes the value w_j in every marking that enables T_j:

$$r(\boldsymbol{m}) \;=\; \begin{cases} w_j & T_j \in E(\boldsymbol{m}) \\ 0 & \mathit{otherwise} \end{cases} \tag{27}$$

The same quantity can also be computed using the more traditional approach of identifying the subset A_j of $RS(\boldsymbol{m}_0)$ in which a given transition T_j is enabled ($A_j \;=\; \{\boldsymbol{m}_i \in RS(\boldsymbol{m}_0) : T_j \in E(\boldsymbol{m}_i)\}$). The mean number of firings of T_j per unit of time is then given by:

$$f_j \;=\; \sum_{\boldsymbol{m}_i \in A_j} w_j \; \pi_i \tag{28}$$

These results show that Petri nets can be used not only as a formalism for describing the behaviour of distributed/parallel systems and for assessing their qualitative properties, but also as a tool for computing performance indices that allow the efficiency of these systems to be evaluated.

To illustrate the details of this last analysis step, a simple example is presented in the following section, which encompasses the explicit derivation of the CTMC infinitesimal generator, of the steady-state probability distribution, and of some performance indices.

An Example SPN Model - Consider a simple shared memory multiprocessor system in which two processors must occasionally access a common shared memory. All the processors are assumed to have identical behaviours characterized by a cyclic sequence of local activities, followed by requests and accesses for the common memory. All these actions last for certain amount of times. Additional delays are experienced by a processor that requests to access the common memory while it is busy serving the other processor. Assuming that all timings considered in the example have negative exponential distributions, the SPN model of this system is depicted in Fig. 19. The net comprises seven places and six timed transitions with single-server semantics.

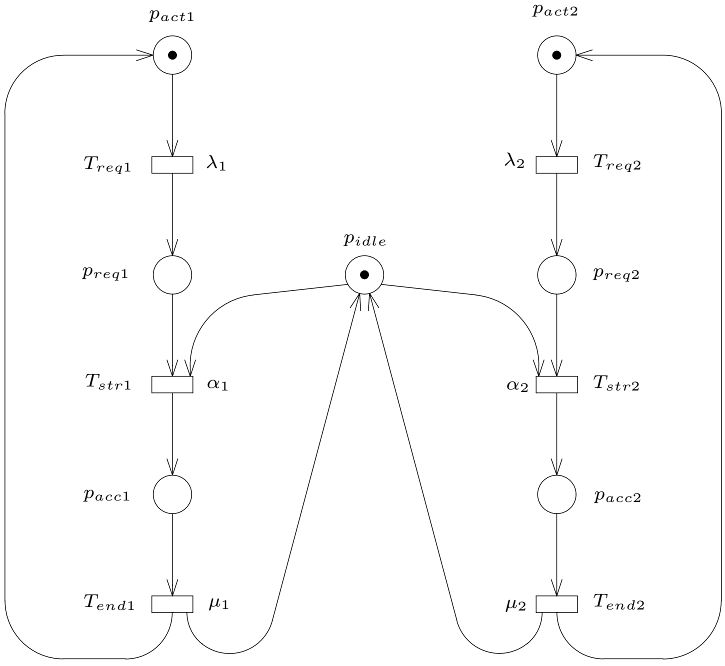


Fig. 19. The SPN description of shared memory system

Starting from the initial marking shown in Fig. 19, in which the two processors are both in a locally active state and the memory is idle ($p_{act1}+p_{act2}+p_{idle}$), a possible evolution of the SPN marking that focuses on the processing cycle of processor 1, may be the following. Processor 1 works locally for an exponentially distributed random amount of time with average $1/\lambda_1$, and then requests an access to the common memory. Transition T_{req1} fires, and the token contained in p_{act1} is removed while a token is added in p_{req1}. Since the common memory is available (place p_{idle} is marked), the acquisition of the memory starts immediately and takes an average of $1/\alpha_1$ units of time to complete; this is represented by the firing of transition T_{str1} whose associated delay has a negative-exponential distribution with rate α_1; when transition T_{str1} fires, one token is removed from place p_{req1} and another token is deposited into place p_{acc1}, where it stays for the entire time required by the first processor to access the common memory. Such a time lasts on the average $1/\mu_1$ units, and ends when transition T_{end1} fires returning the net to its initial state. Obviously, a similar processing cycle is possible for processor 2 and many interleavings between the activities of the two processors may be described by the evolution of the net.

A conflict exists in the behaviour of this system when both processors want to simultaneously access the common memory, i.e., when transitions T_{str1} and T_{str2} are both enabled. According to Equation (13), in this situation, transition T_{str1} fires with probability:

$$P\{T_{str1}\} = \frac{\alpha_1}{\alpha_1 + \alpha_2} \tag{29}$$

whereas transition T_{str2} fires with probability:

$$P\{T_{str2}\} = \frac{\alpha_2}{\alpha_1 + \alpha_2} \tag{30}$$

Notice that when the two transitions T_{str1} and T_{str2} are both enabled in a given marking M, the speed at which the PN model exits from that marking is the sum of the individual speeds of the two transitions and the conflict is actually resolved only at the moment the first of them fires.

Table 1. Reachability set of SPN of Fig. 19

$\boldsymbol{m}_0 =$	p_{act1} +			p_{idle} +	p_{act2}		
$\boldsymbol{m}_1 =$		p_{req1} +		p_{idle} +	p_{act2}		
$\boldsymbol{m}_2 =$			p_{acc1} +		p_{act2}		
$\boldsymbol{m}_3 =$			p_{acc1} +			p_{req2}	
$\boldsymbol{m}_4 =$		p_{req1} +		p_{idle} +		p_{req2}	
$\boldsymbol{m}_5 =$	p_{act1} +			p_{idle} +		p_{req2}	
$\boldsymbol{m}_6 =$	p_{act1} +						p_{acc2}
$\boldsymbol{m}_7 =$		p_{req1} +					p_{acc2}

The reachability set of the SPN model of Fig. 19 is listed in Table 1. The reachability graph is shown in Fig. 20, while the corresponding CTMC transition rate diagram is presented in Fig. 21.

8 Generalized Stochastic Petri Nets

Several reasons suggest the introduction of the possibility of using immediate transitions into PN models together with timed transitions. As we observed in

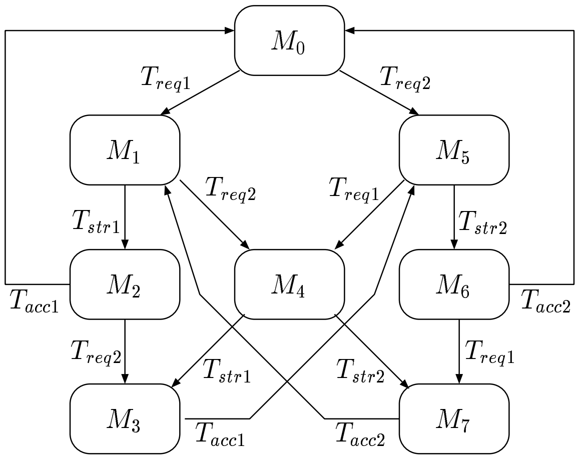


Fig. 20. The reachability graph of the SPN system in Fig. 19

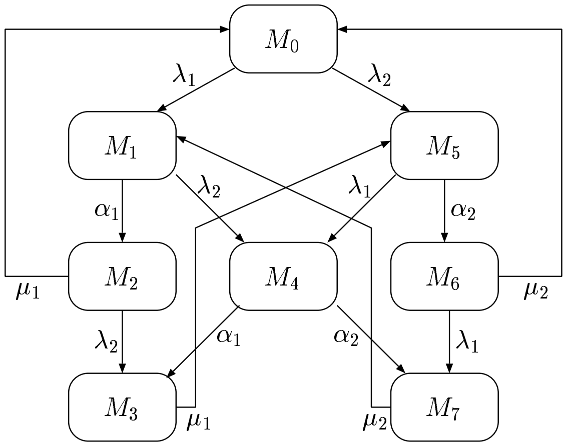


Fig. 21. The state transition rate diagram of the Markov chain associated with the SPN in Fig. 19

Section 6.3 the firing of a transition may describe either the completion of a time-consuming activity, or the verification of a logical condition. It is thus natural to use timed transitions in the former case, and immediate transitions in the latter. Moreover, as we noted in the previous sections, when all transitions are timed the temporal specification of the model must in some cases consider at one time both the timing and the probability inherent in a choice. It seems natural to separate the two aspects in the modelling paradigm, to simplify the model specification. Furthermore, by allowing the use of immediate transitions, some important benefits can be obtained in the model solution. They will be described in detail later in this section; we only mention here the fact that the use of immediate transitions may significantly reduce the cardinality of the reachability set, and may eliminate the problems due to the presence in the model of timed transitions with rates that differ by orders of magnitude. The latter situation results in so-called "stiff" stochastic processes, that are quite difficult to handle from a numerical viewpoint. On the other hand, the introduction of immediate transitions in an SPN does not raise any significant complexity in the analysis, as we shall see soon.

SPN models in which immediate transitions coexist with timed transitions with race policy and random firing delays with negative exponential pdf are known by the name *generalized SPNs* (GSPNs) [4].

In the graphical representation of GSPNs, immediate transitions are drawn as bars or segments and are denoted by a name that is normally of the form t_x, where x is either a number or a mnemonic string; timed transitions are drawn as (white or black) rectangular boxes, and are denoted by a name that is normally of the form T_x.

Immediate transitions are fired with priority over timed transitions. Thus, if timing is disregarded, the resulting PN model comprises transitions at different priority levels. The adoption of the race policy may seem to implicitly provide the priority of immediate over timed transitions; this is indeed the case in most situations, but the explicit use of priority simplifies the development of the theory. We shall return to this subtle point later in this section.

Recall that markings in the reachability set can be classified as *tangible* or *vanishing.* A marking in which no transition is enabled is tangible. The time spent in any vanishing marking is deterministically equal to zero, while the time spent in tangible markings is positive with probability one.

To describe the GSPN dynamics, we separately observe the timed and the immediate behaviour, hence referring to tangible and vanishing markings, respectively. Let us start with the timed dynamics (hence with tangible markings); this is identical to the dynamics in SPNs, that was described before. We can assume that each timed transition possesses a timer. The timer is set to a value that is sampled from the negative exponential pdf associated with the transition, when the transition becomes enabled for the first time after firing. During all time intervals in which the transition is enabled, the timer is decremented. Transitions fire when their timer reading goes down to zero.

With this interpretation, each timed transition can be used to model the execution of some activity in a distributed environment; all enabled activities execute in parallel (unless otherwise specified by the PN structure) until they complete. At completion time, activities induce a change of the system state, only as regards their local environment. No special mechanism is necessary for the resolution of timed conflicts: the temporal information provides a metric that allows the conflict resolution.

In the case of vanishing markings, the GSPN dynamics consumes no time: everything takes place instantaneously. This means that if only one immediate transition is enabled, it fires, and the following marking is produced. If several immediate transitions are enabled, a metric is necessary to identify which transition will produce the marking modification. Actually, the selection of the transition to be fired is relevant only in those cases in which a conflict must be resolved: if the enabled transitions are concurrent, they can be fired in any order. For this reason, GSPNs associate *weights* with immediate transitions belonging to the same conflict set.

For the time being, let us consider only free-choice conflict sets; the case of non-free-choice conflict sets will be considered later on, but we can anticipate at this point that it can be tackled in a similar manner by exploiting the definition of ECS introduced in Section 5.1. The transition weights are used to compute the firing probabilities of the simultaneously enabled transitions comprised within the conflict set. The restriction to free-choice conflict sets guarantees that transitions belonging to different conflict sets cannot disable each other, so that the selection among transitions belonging to different conflict sets is not necessary.

We can thus observe a difference between the specification of the temporal information for timed transitions and the specification of weights for immediate transitions. The temporal information associated with a timed transition depends only on the characteristics of the activity modelled by the transition. Thus, the temporal specification of a timed transition requires no information on the other (possibly conflicting) timed transitions, or on their temporal characteristics. On the contrary, for immediate transitions, the specification of weights must be performed considering at one time all transitions belonging to the same conflict set. Indeed, weights are normalized to produce probabilities by considering all enabled transitions within a conflict set, so that the specification of a weight, independent of those of the other transitions in the same conflict set, is not possible.

8.1 Some Extensions

Some additional features can be included in a GSPN model, with an advantage in the power of the modelling paradigm and little increase in the analysis complexity. These extensions are the possibility of using non-free-choice conflicts of immediate transitions, the availability of multiple priority levels for immediate transitions, and the marking-dependency of transition annotations (rates for timed transitions and weights for immediate transitions).

The availability of multiple priority levels for immediate transitions, and the marking-dependency of transition annotations has quite a beneficial impact on the modelling power, and hardly any impact on the complexity of the model specification and analysis. In particular, the availability of multiple priority levels for immediate transitions permits the simple specification of complex sequential selection algorithms, while the marking-dependency of transition annotations allows the development of compact models in which the behaviours of a number of different entities are synthetically described.

Employing non-free-choice conflicts of immediate transitions, the user has the possibility of describing a much wider range of dynamic behaviours in vanishing markings, but he must be able to correctly associate the immediate transitions with the metrics that define the probabilistic conflict resolution. This requires the knowledge of the sets of simultaneously enabled non-concurrent immediate transitions in any vanishing marking. This knowledge may not be easy to obtain without the generation of the reachability set, which is however very costly in most cases. The definition of extended conflict sets (ECSs) was introduced in Section 5 to provide the user with the information on the sets of transitions that *may* be in effective conflict (either direct or indirect) in a marking, thus helping in the definition of weights.

One extension that is not possible within the GSPN framework is the introduction of more general forms of pdf for the firing delays associated with transitions. Nevertheless, also in this respect, the availability of immediate and exponential transitions in one modelling paradigm can be exploited for the construction of somewhat more general pdfs in the description of the duration of the real system activities (see [6] for a detailed discussion of this topic).

8.2 The Definition of a GSPN Model

GSPNs were originally defined in [4]. The definition was later improved to better exploit the structural properties of the modelling paradigm [3]. The definition we present here is based on the version contained in this second proposal.

Formally, a GSPN model is an 8-tuple

$$\text{GSPN} = (P, T, \Pi(.), I(.), O(.), H(.), W(.), \boldsymbol{m}_0) \tag{31}$$

where $\text{PN}_\pi = (P, T, \Pi(.), I(.), O(.), H(.), \boldsymbol{m}_0)$ is the marked PN with priority underlying the GSPN and $W(.)$ is a function defined on the set of transitions

Timed transitions are associated with priority zero, whereas all other priority levels are reserved for immediate transitions.

The underlying PN model constitutes the structural component of a GSPN model, and it must be confusion-free (see Section 5.1) at priority levels greater than zero (i.e., in subnets of immediate transitions).

The function W allows the definition of the stochastic component of a GSPN model. In particular, it maps transitions into real positive functions of the GSPN marking. Thus, for any transition t it is necessary to specify a function $W(t, \boldsymbol{m})$. In the case of marking independency, the simpler notation w_k is normally used to indicate $W(t_k)$, for any transition $t_k \in T$. The quantity $W(t_k, \boldsymbol{m})$ (or w_k) is called the "rate" of transition t_k in marking $\boldsymbol{m}$ if t_k is timed, and the "weight" of transition t_k in marking M if t_k is immediate.

Since in any marking all firing delays of timed transitions have a negative exponential pdf, and all the delays are independent random variables, the sojourn time in a tangible marking is a random variable with a negative exponential pdf whose rate is the sum of the rates of all enabled timed transitions in that marking. In the case of vanishing markings, the weights of the immediate transitions enabled in an ECS can be used to determine which immediate transition will actually fire, if the vanishing marking enables more than one conflicting immediate transition.

When transitions belonging to several different *ECS*s are simultaneously enabled in a vanishing marking, as we already explained, the choice among these transitions is irrelevant. It must be emphasized that the irrelevance in the order of transition firings is an important consequence of the restriction that subnets of immediate transitions must be confusion-free. The restriction to confusion-free immediate subnets also has a beneficial impact on the model definition, since the association of weights with immediate transitions requires only the information about *ECS*s, not about reachable markings. For each *ECS* the analyst thus defines a *local* association of weights from which probabilities are then derived.

8.3 Some Fine Points

Let us return to the points we raised in the previous sections, but left unanswered. These are:

1. the need for an explicit priority of immediate over timed transitions;
2. the irrelevance of the distinction between resampling, enabling memory, and age memory;
3. the impossibility for two timers to expire at the same time.

All three points relate to the properties of the negative exponential pdf. This distribution characterizes a continuous random variable, hence it is a continuous function defined in the interval $[0, \infty)$, that integrates to one. The lack of discontinuities in the function makes the probability of any specific value x being sampled equal to zero (however, obviously, the probability that a value is sampled between two distinct values $x_1 \geq 0$ and $x_2 > x_1$ is positive).

Let us look now at the three previous points in order.

1. Consider a free-choice conflict set comprising an immediate and a timed transition, and assume for a moment that priority does not exist. The race policy makes the immediate transition always win, except for the case in which a zero delay is sampled from the negative exponential pdf. Although the probability of selecting the value zero is null, some problem may arise when the conflict set is enabled infinitely often in a finite time interval. For example, in the case of Fig. 22, the timed and the immediate transitions are always enabled, because the firing of the immediate transition t_2 does not alter the PN marking. The situation is changed only when the timed transition T_1 fires. This happens with probability one in time zero, possibly after an infinite number of firings of the immediate transition. To avoid these (sometimes strange) limiting behaviours, the priority of immediate over timed transitions was introduced in the GSPN definition. This makes the timed transition T_1 in Fig. 22 never enabled.
2. The *memoryless property* of the negative exponential pdf, ensures that at any time instant, the residual time until a timer associated with a transition expires is statistically equivalent to the originally sampled timer reading. Thus, whether a new timer value is set at every change of marking, or at every instant a transition becomes enabled after disabling, or after firing,

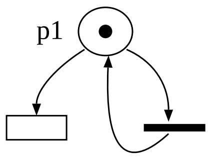


Fig. 22. A conflict set comprising a timed and an immediate transition

makes no difference from the point of view of the probabilistic metrics of the GSPN.

3. Since the probability that a sample extracted from a negative exponential pdf takes a specific value x equals zero, the probability of two timers expiring at the same time is null. Indeed, given the value sampled by the first timer, the probability that the second one samples the same value is zero.

8.4 The Stochastic Process Associated with a GSPN

As we have just observed, GSPNs adopt the same firing policy of SPNs; when several transitions are enabled in the same marking, the probabilistic choice of the transition to fire next depends on parameters that are associated with these same transitions and that are not functions of time. The general expression for the probability that a given (timed or immediate) transition t_k, enabled in marking $\boldsymbol{m}_i$, fires is:

$$P\{t_k|\boldsymbol{m}_i\} \;=\; \frac{w_k}{q_i} \tag{32}$$

where q_i is the quantity defined by Equation (14). Equation (32) represents the probability that transition t_k fires first, and is identical to Equation (20) for SPNs, with a difference in the meaning of the parameters w_k. When the marking is vanishing, the parameters w_k are the weights of the immediate transitions enabled in that marking and define the selection policy used to make the choice. When the marking is tangible, the parameters w_k of the timed transitions enabled in that marking are the rates of their associated negative exponential distributions. The average sojourn time in vanishing markings is zero, while the average sojourn time in tangible markings is given by Equation (21).

Observing the evolution of the GSPN system, we can notice that the distribution of the sojourn time in an arbitrary marking can be expressed as a composition of negative exponential and deterministically zero distributions: we can thus recognize that the marking process $\{\mathcal{M}(\tau), \tau \geq 0\}$ is a semi-Markov stochastic process.

When several immediate transitions are enabled in the same vanishing marking, deciding which transition to fire first makes sense only in the case of conflicts. If these immediate transitions do not "interfere" they could be fired simultaneously and the choice of firing only one of them at a time becomes an operational rule of the model that hardly relates with the actual characteristics of the DEDS we are modelling. In this case the selection is inessential from the point of view of the overall behaviour of the net.

Assuming that the GSPN is not confused, the computation of the ECSs of the net corresponds to partitioning the set of immediate transitions into equivalence classes such that transitions of the same partition may be in conflict among each other in possible markings of the net, while transitions of different ECSs behave in a truly concurrent manner.

When transitions belonging to the same ECS are the only ones enabled in a given marking, one of them (say transition t_k) is selected to fire with probability:

$$P\{t_k|\boldsymbol{m}_i\} \;=\; \frac{w_k}{\omega_k(\boldsymbol{m}_i)} \tag{33}$$

where $\omega_k(\boldsymbol{m}_i)$ is the weight of ECS(t_k) in marking $\boldsymbol{m}_i$ and is defined as follows:

$$\omega_k(\boldsymbol{m}_i) \;=\; \sum_{t_j \in [\mathrm{ECS}(t_k) \wedge E(\boldsymbol{m}_i)]} w_j \tag{34}$$

Within the ECS we may have transitions that are in direct as well as in indirect conflicts. This means that the firing selection probabilities may be different for the same transition in different markings. Equation (33) however ensures that if we have two transitions (say transitions t_i and t_j), both enabled in two different markings (say markings $\boldsymbol{m}_r$ and $\boldsymbol{m}_s$), the ratios between the firing probabilities of these two transitions in these two markings remain constant and in particular equal to the ratio between the corresponding weights assigned at the moment of the specification of the model.

During the evolution of a GSPN, it may happen that several ECSs are simultaneously enabled in a vanishing marking. According to the usual firing mechanism of Petri nets, we should select the transition to fire by first non-deterministically choosing one of the ECSs and then a transition within it. The assumption that the GSPN is not confused guarantees that the way in which the choice of the ECS is performed is irrelevant with respect to the associated stochastic process. One possibility is that of computing the weight of the ECS by adding the parameters of all the enabled transitions that belong to that ECS and of using this weight to select the ECS with a method suggested by Equation (32). A simple derivation shows that the use of this method implies that the selection of the immediate transition to be fired in a vanishing marking can be performed with the general formula (32) that was originally derived for timed transitions (and thus for tangible markings) only [3,21]. Moreover, it is possible to show that if we consider the probabilities associated with the many different sequences of immediate transitions whose firings lead from a given vanishing marking to a target tangible one, they turn out to be all equal [3].

This last property strongly depends on the absence of confusion in the GSPN; the fact that the presence of confused subnets of immediate transitions within a GSPN is an undesirable feature of the model can also be explained considering the following example.

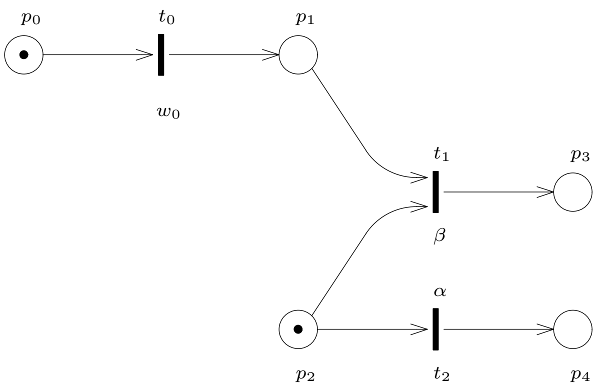


Fig. 23. Example of a simple confused GSPN model

Suppose that a subnet is identified as confused using the structural confusion condition of Section 5.1. Suppose also that the subnet has the structure depicted in Fig. 23. Given the initial marking $\boldsymbol{m}_0 = (p_0 + p_2)$, markings $\boldsymbol{m}_1 = p_3$ and $\boldsymbol{m}_2 = (p_1 + p_4)$ are reached with total probabilities

$$P\{\boldsymbol{m}_0, \boldsymbol{m}_1\} = \frac{w_0}{(w_0 + \alpha)} \frac{\beta}{(\alpha + \beta)} \qquad P\{\boldsymbol{m}_0, \boldsymbol{m}_2\} = \frac{\alpha}{(w_0 + \alpha)} \quad .$$

From these expressions we can see that, although the picture suggests transitions t_0 and t_2 as concurrent, and transitions t_1 and t_2 as members of a conflict set, the first choice of firing either t_0 or t_2 is actually crucial for the possibility of reaching the desired markings. In this case the value assigned by the analyst to w_0 becomes fundamental for the quantitative evaluation of the model.

Much more intriguing however is the behaviour of the subnet in Fig. 24(b) that differs from that of Fig. 24(a) for the simple addition of place p_a and transition t_a between transition t_0 and place p_1. Given the initial marking $\boldsymbol{m}_0 = (p_0 + p_2)$, lmarkings $\boldsymbol{m}_1 = p_3$ and $\boldsymbol{m}_2 = (p_1 + p_4)$ are reached in the case of the subnet of Fig. 24(b) with total probabilities

$$P\{\boldsymbol{m}_0, \boldsymbol{m}_1\} = \frac{w_0}{(w_0 + \alpha)} \frac{w_a}{(w_a + \alpha)} \frac{\beta}{(\alpha + \beta)}$$

and

$$P\{\boldsymbol{m}_0, \boldsymbol{m}_2\} = \frac{\alpha}{(w_0 + \alpha)} + \frac{w_0}{(w_0 + \alpha)} \frac{\alpha}{(w_a + \alpha)} \quad .$$

From a modelling point of view, there are good reasons to consider the two subnets of Figs. 24(a) and 24(b) equivalent since the sequence of immediate actions represented by t_0 and t_a of Fig. 24(b) should be reducible to transition t_0 of Fig. 24(a) without affecting the behaviour of the model. Instead, the difference among the total probabilities that we have just computed shows that, in the case of confused models, the trivial action of splitting an atomic (and instantaneous) action in two has drastic effects not only on the graphical description of the

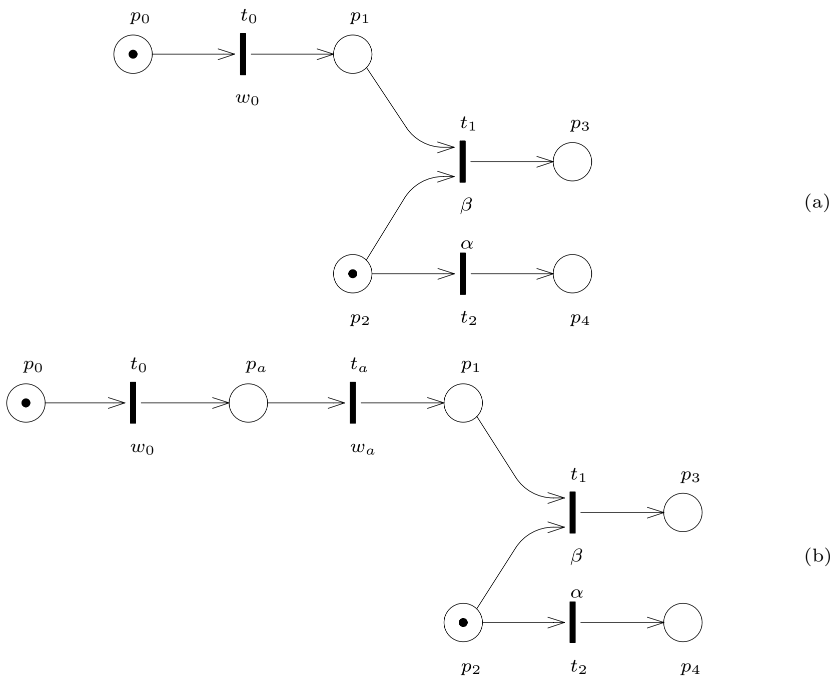


Fig. 24. Comparison of two simple confused GSPN systems

model, but also (and more important) on the values of the results obtained from its quantitative evaluation.

Marking Dependency - The firing times and the weights that we have considered so far were assumed to be independent of the marking of the GSPN. In principle, however, it is possible to work with transition parameters that are marking-dependent as pointed out at the beginning of this section. When one or more timed transitions are enabled in a given marking, we can compute the distribution of the sojourn time in that marking, as well as the probability of the transition that fires first, using negative exponential distributions whose firing rates may depend on that specific marking. Similarly, the selection of the immediate transition that fires in a vanishing marking enabling several immediate transitions can be computed using weighting factors that may be marking-dependent. In all these cases, Equations (21), (32), (33), and (34) can be generalized assuming that all the parameters are functions of the marking for which they are computed.

In practice, the generality allowed by this extension (which was assumed by the original GSPN proposal [4]) is in contrast with the claim of GSPNs as a high-level language for the description of complex systems. In fact, while the construction of a GSPN model requires a local view of the behaviour of the real system, the specification of (general) marking-dependent parameters requires the analyst to be aware of the possible (global) states of the system. Moreover, a dependency of the weights of conflicting immediate transitions on the marking of places that are not part of the input set of any of the transitions comprised in their ECS could lead to a new form of "confusion" that is not captured by the definitions contained in Section 5.1 and discussed in the previous section.

For this reason, a restriction of the type of marking-dependency allowed in GSPN models was informally proposed in [21]. A definition of marking dependency that satisfies these restrictions can be obtained by allowing the specification of marking dependent parameters as the product of a nominal rate (or weight in the case of immediate transitions) and of a dependency function defined in terms of the marking of the places that are connected to a transition through its input and inhibition functions.

Denote with $\boldsymbol{m}_{/t}$ the restriction of a generic marking M to the input and inhibition sets of transition t:

$$\boldsymbol{m}_{/t} \;=\; \bigcup\nolimits_{p \in (^\bullet t \cup {}^\circ t)} m(p) \tag{35}$$

Let $f(\boldsymbol{m}_{/t})$ be the marking dependency function that assumes positive values every time transition t is enabled in M; using this notation, the marking dependent parameters may be defined in the following manner:

$$\begin{cases} \mu_i(\boldsymbol{m}) \;=\; f(\boldsymbol{m}_{/T_i})\; w_i & \text{in case of firing rates,} \\ \\ \omega_j(\boldsymbol{m}) \;=\; f(\boldsymbol{m}_{/t_j})\; w_j & \text{in case of weights.} \end{cases} \tag{36}$$

Multiple-servers and infinite-servers can be represented as special cases of timed transitions with marking dependent firing rates that are consistent with this restriction. In particular, a timed transition T_i with a negative-exponential delay distribution with parameter w_i and with an infinite-server policy, has a marking dependency function of the following form:

$$f(\boldsymbol{m}_{/T_i}) \;=\; \boldsymbol{e}_i(\boldsymbol{m}) \tag{37}$$

where $\boldsymbol{e}_i(\boldsymbol{m})$ is the enabling degree of transition T_i in marking $\boldsymbol{m}$. Similarly, a timed transition T_i with multiple-server policy of degree K has a marking dependency function defined in the following way:

$$f(\boldsymbol{m}_{/T_i}) \;=\; min\;(\boldsymbol{e}_i(\boldsymbol{m}),\; K) \tag{38}$$

Other interesting situations can be represented using the same technique, Fig. 25 depicts two such cases. In Fig. 25(a), we have a situation of two competing infinite servers: transition T_1 fires with rate $w_1 m(p_1)$ if place p_0 is marked; similarly for transition T_2. Obviously both transitions are interrupted when the first of the two fires removing the token from place p_0.

Using $\delta(x)$ to represent the following step function:

$$\delta(x) \;=\; \begin{cases} 0 & x \;=\; 0 \\ \\ 1 & x \;>\; 0 \end{cases} \tag{39}$$

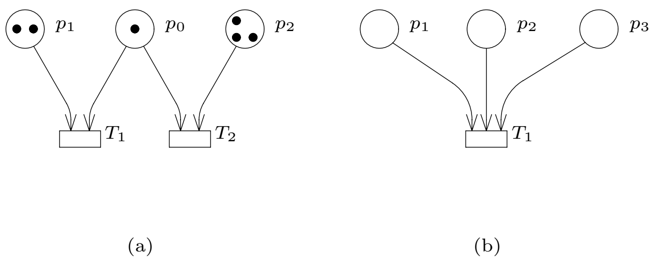


Fig. 25. Examples of complex marking dependency situations

we can define the marking dependency function as follows:

$$\begin{cases} f(\boldsymbol{m}_{/T_1}) = \delta(ED(T_1,\boldsymbol{m}))m(p_1) \\ \qquad \text{and similarly} \\ f(\boldsymbol{m}_{/T_2}) = \delta(ED(T_2,\boldsymbol{m}))m(p_2) \end{cases} \tag{40}$$

Fig. 25(b) represents instead a server whose speed depends on a linear combination of the markings of all its input places. In this case the marking dependency function may assume the form:

$$f(\boldsymbol{m}_{/T_1}) = \delta(\boldsymbol{e}_1(\boldsymbol{m})) \sum_{p \in {}^\bullet T_1} \alpha_p m(p) \qquad \alpha_p \geq 0, \sum_{p \in {}^\bullet T_1} \alpha_p = 1 \tag{41}$$

8.5 Numerical Solution of GSPN Systems

The stochastic process associated with a k-bounded GSPN system with $\boldsymbol{m}_0$ as its home state can be classified as a finite state space, stationary (homogeneous), irreducible, and continuous-time semi-Markov process.

Semi-Markov processes can be analysed identifying an embedded (discrete-time) Markov chain that describes the transitions from state to state of the process. In the case of GSPNs, the embedded Markov chain (EMC) can be recognized disregarding the concept of time and focusing the attention on the set of states of the semi-Markov process. The specifications of a GSPN are sufficient for the computation of the transition probabilities of such a chain.

Let RS, TS, and VS indicate the state space (the reachability set), the set of tangible states (or markings) and the set of vanishing markings of the stochastic process, respectively. The following relations hold among these sets:

$$RS = TS \bigcup VS, \qquad TS \bigcap VS = \emptyset.$$

The transition probability matrix $\boldsymbol{U}$ of the EMC can be obtained from the specification of the model using the following expression:

$$u_{ij} = \frac{\sum_{T_k \in E_j(\boldsymbol{m}_i)} w_k}{q_i} \tag{42}$$

n this way, except for the diagonal elements of matrix $\boldsymbol{U}$, ll the other transition probabilities of the EMC an be computed using quation (32) independently of whether the transition to be onsidered is timed or immediate, according to the discussion contained t the end of Section 8.4.

By ordering the markings so that the vanishing ones correspond to the first entries of the matrix and the tangible ones to the last, the transition probability matrix $\boldsymbol{U}$ can be decomposed in the following manner:

$$\boldsymbol{U} \; = \; \boldsymbol{A} + \boldsymbol{B} \; = \; \begin{bmatrix} \boldsymbol{C} & \boldsymbol{D} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \boldsymbol{E} & \boldsymbol{F} \end{bmatrix} \tag{43}$$

The elements of matrix $\boldsymbol{A}$ correspond to changes of markings induced by the firing of immediate transitions; in particular, those of submatrix $\boldsymbol{C}$ are the probabilities of moving from vanishing to vanishing markings, while those of $\boldsymbol{D}$ correspond to transitions from vanishing to tangible markings. Similarly, the elements of matrix $\boldsymbol{B}$ correspond to changes of markings caused by the firing of timed transitions: $\boldsymbol{E}$ accounts for the probabilities of moving from tangible to vanishing markings, while $\boldsymbol{F}$ comprises the probabilities of remaining within tangible markings.

Indicating with $\boldsymbol{\psi}(n)$ the probability distribution of the EMC at step n (i.e., after n (state-) transitions performed by the EMC), we can compute this quantity using the following expression

$$\boldsymbol{\psi}(n) \; = \; \boldsymbol{\psi}(0)\boldsymbol{U}^n \tag{44}$$

where, as usual, $\boldsymbol{\psi}(0)$ represents the initial distribution of the EMC. The steady-state probability distribution $\boldsymbol{\psi}$ can be obtained as the solution of the system of linear equations

$$\begin{cases} \boldsymbol{\psi} \; = \; \boldsymbol{\psi}\,\boldsymbol{U} \\ \boldsymbol{\psi}\,\mathbf{1}^{\mathrm{T}} \; = \; 1 \end{cases} \tag{45}$$

The steady-state probability distribution of the EMC, can be interpreted in terms of numbers of (state-) transitions performed by the EMC. In fact, $1/\psi_i$ is the mean recurrence time for state s_i (marking $\boldsymbol{m}_i$) measured in number of transition firings. The steady-state probability distribution of the stochastic process associated with the GSPN system is thus obtained by weighting each entry ψ_i with the sojourn time of its corresponding marking SJ_i and by normalizing the whole distribution.

The solution method outlined so far, is computationally acceptable whenever the size of the set of vanishing markings is small (compared with the size of the set of tangible markings). However, this method requires the computation of the steady-state probability of each vanishing marking that is known a priori to be null. Moreover, vanishing markings, by enlarging the size of the transition probability matrix $\boldsymbol{U}$, tend to make the solution more expensive and in some cases even impossible to obtain.

In order to restrict the solution to quantities directly related with the computation of the transient and steady-state probabilities of tangible markings, we must reduce the model by computing the total transition probabilities among tangible markings only, thus identifying a Reduced EMC (REMC).

To illustrate the method of reducing the EMC by removing the vanishing markings, consider first the example of Fig. 26. This system contains two free-choice conflicts corresponding to transitions T_1 , T_2, and t_1, t_2, respectively. From the initial marking $\boldsymbol{m}_i = p_1$, the system can move to marking $\boldsymbol{m}_j = p_3$ following two different paths. The first corresponds to the firing of transition T_1, that happens with probability $\frac{\mu_1}{(\mu_1+\mu_2)}$, and that leads to the desired (target) marking $\boldsymbol{m}_j$ in one step only. The second corresponds to selecting transition T_2 to fire first, followed by transition t_1. The first of these two events happens with probability $\frac{\mu_2}{(\mu_1+\mu_2)}$, and the second with probability $\frac{\alpha}{(\alpha+\beta)}$. The total probability of this second path from $\boldsymbol{m}_i$ to $\boldsymbol{m}_j$ amounts to $\frac{\mu_2}{(\mu_1+\mu_2)}\frac{\alpha}{(\alpha+\beta)}$. Notice that firing transition T_2 followed by transition t_2 would lead to a different marking (in this case the initial one). Firing transition T_2 leads the system into an intermediate (vanishing) marking $\boldsymbol{m}_r$. The total probability of moving from marking $\boldsymbol{m}_i$ to marking $\boldsymbol{m}_j$ is thus in this case:

$$u'_{ij} = \frac{\mu_1}{(\mu_1+\mu_2)} + \frac{\mu_2}{(\mu_1+\mu_2)}\frac{\alpha}{(\alpha+\beta)} \tag{46}$$

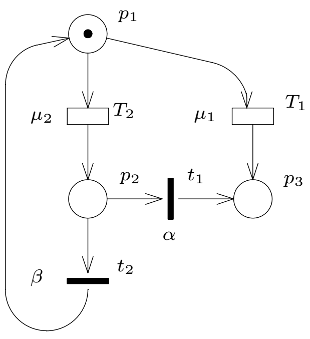


Fig. 26. A GSPN system with multiple paths between tangible markings

In general, upon the exit from a tangible marking, the system may "walk" through several vanishing markings before ending in a tangible one. To make the example more interesting, and to illustrate more complex cases, let us modify the net of Fig. 26 as depicted in Fig. 27. In this new case the system can move from marking $\boldsymbol{m}_1 = p_1$ to marking $\boldsymbol{m}_2 = p_2$ following four different paths. The first corresponds again to firing transition T_1 and thus to a direct move from the initial marking to the target one. This happens with probability $\frac{\mu_1}{(\mu_1+\mu_2+\mu_3)}$. The second path corresponds to firing transition T_2 followed by t_1 (total probability $\frac{\mu_2}{(\mu_1+\mu_2+\mu_3)} \frac{\alpha}{(\alpha+\beta)}$); the third path corresponds to firing transition T_3 followed by transition t_4 (total probability $\frac{\mu_3}{(\mu_1+\mu_2+\mu_3)} \frac{\gamma}{(\gamma+\delta)}$). Finally, the last path corresponds to firing transition T_3 followed by transition t_3 and then by transition t_1 which happens with probability $\frac{\mu_3}{(\mu_1+\mu_2+\mu_3)} \frac{\delta}{(\gamma+\delta)} \frac{\alpha}{(\alpha+\beta)}$.

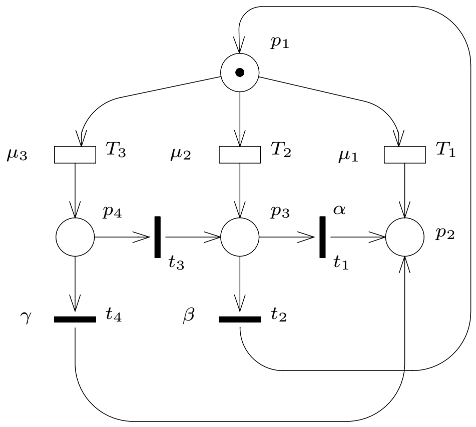


Fig. 27. A GSPN system with several multiple paths between tangible markings

In this case the total probability of moving from marking $\boldsymbol{m}_i$ to marking $\boldsymbol{m}_j$ becomes:

$$\begin{aligned} u'_{ij} \;=\; & \frac{\mu_1}{(\mu_1+\mu_2+\mu_3)} + \frac{\mu_2}{(\mu_1+\mu_2+\mu_3)} \frac{\alpha}{(\alpha+\beta)} + \\ & \frac{\mu_3}{(\mu_1+\mu_2+\mu_3)} \frac{\gamma}{(\gamma+\delta)} + \frac{\mu_3}{(\mu_1+\mu_2+\mu_3)} \frac{\alpha}{(\alpha+\beta)} \frac{\delta}{(\gamma+\delta)} \end{aligned} \tag{47}$$

In general, recalling the structure of the $\boldsymbol{U}$ matrix, a direct move from marking $\boldsymbol{m}_i$ to marking $\boldsymbol{m}_j$ corresponds to a non-zero entry in block $\boldsymbol{F}$ ($f_{ij} \neq 0$), while a path from $\boldsymbol{m}_i$ to $\boldsymbol{m}_j$ via two intermediate vanishing markings corresponds to the existence of

1. a non-zero entry in block $\boldsymbol{E}$ corresponding to a move from $\boldsymbol{m}_i$ to a generic intermediate marking $\boldsymbol{m}_r$;
2. a non-zero entry in block $\boldsymbol{C}$ from this generic state $\boldsymbol{m}_r$ to another arbitrary vanishing marking $\boldsymbol{m}_s$;
3. a corresponding non-zero entry in block $\boldsymbol{D}$ from $\boldsymbol{m}_s$ to $\boldsymbol{m}_j$.

These informal considerations are precisely captured by the following formula:

$$u'_{ij} \;=\; f_{ij} \;+ \sum_{r:\boldsymbol{m}_r \in VS} e_{ir} P\{r \rightarrow s\}\, d_{sj} \tag{48}$$

where $P\{r \rightarrow s\}\, d_{sj}$ is the probability that the net moves from vanishing marking $\boldsymbol{m}_r$ to tangible marking $\boldsymbol{m}_j$ in an arbitrary number of steps, following a path through vanishing markings only.

In order to provide a general and efficient method for the computation of the state transition probability matrix $\boldsymbol{U}'$ of the REMC, we can observe that Equation (48) can be rewritten in matrix notation in the following form:

$$\boldsymbol{U}' \;=\; \boldsymbol{F} \;+\; \boldsymbol{E}\,\boldsymbol{G}\,\boldsymbol{D} \tag{49}$$

where each entry g_{rs} of matrix $\boldsymbol{G}$ represents the probability of moving from vanishing marking $\boldsymbol{m}_r$ to vanishing marking $\boldsymbol{m}_s$ in any number of steps, but without hitting any intermediate tangible marking. $\boldsymbol{G}$ can be expressed with the following formula:

$$\boldsymbol{G} \;=\; \sum_{n=0}^{\infty} \boldsymbol{C}^n$$

In the computation of $\boldsymbol{C}^n$, two possibilities may arise. The first corresponds to the situation in which there are no loops among vanishing markings. This means that for any vanishing marking $\boldsymbol{m}_r \in VS$ there is a value n_{0r} such that any sequence of transition firings of length $n \geq n_{0r}$ starting from such marking must reach a tangible marking $\boldsymbol{m}_j \in TS$. In this case

$$\exists n_0 \;:\quad \forall\; n \geq n_0 \quad \boldsymbol{C}^n \;=\; 0$$

and

$$\boldsymbol{G} \;=\; \sum_{k=0}^{\infty} \boldsymbol{C}^k \;=\; \sum_{k=0}^{n_0} \boldsymbol{C}^k$$

The second corresponds to the situation in which there are possibilities of loops among vanishing markings, so that the GSPN may remain "trapped" within a set of vanishing markings. In this case the irreducibility property of the semi-Markov process associated with the GSPN system ensures that the following results hold [69]:

$$\lim_{n \rightarrow \infty} \boldsymbol{C}^n \;=\; 0$$

so that

$$\boldsymbol{G} \;=\; \sum_{k=0}^{\infty} \boldsymbol{C}^k \;=\; [\boldsymbol{I} \;-\; \boldsymbol{C}]^{-1}.$$

We can thus write (see [5,4] for details):

$$\boldsymbol{H} \;=\; \begin{cases} \left(\sum_{k=0}^{n_0} \boldsymbol{C}^k \right) \boldsymbol{D} & \textit{no loops among vanishing states} \\[2ex] [\boldsymbol{I} \;-\; \boldsymbol{C}]^{-1}\, \boldsymbol{D} & \textit{loops among vanishing states} \end{cases}$$

from which we can conclude that an explicit expression for the desired total transition probability among any two tangible markings is:

$$u'_{ij} \;=\; f_{ij} \;+\; \sum_{r \in VS} e_{ir}\, h_{rj} \quad \forall \quad i,j \in TS$$

The transition probability matrix of the REMC can thus be expressed as

$$\boldsymbol{U}' = \boldsymbol{F} + \boldsymbol{E}\,\boldsymbol{H} \tag{50}$$

Denoting again with $\boldsymbol{\psi'}(n)$ the probability distribution of the REMC at step n (i.e., after the firing of n timed transitions), we can compute this quantity using the following expression

$$\boldsymbol{\psi}'(n) = \boldsymbol{\psi}'(0)\boldsymbol{U}'^{n} \tag{51}$$

The steady-state probability distribution $\boldsymbol{\psi}'$ can be obtained as the solution of the system of linear equations

$$\begin{cases} \boldsymbol{\psi'} = \boldsymbol{\psi'}\,\boldsymbol{U'} \\ \\ \boldsymbol{\psi'}\,\mathbf{1}^{\mathrm{T}} = 1. \end{cases} \tag{52}$$

The stationary probability distribution associated with the set of tangible markings is thus readily obtained by means of their average sojourn times (see Equation (21)) using a procedure similar vto that outlined before for the EMC method.

The construction of the REMC defined over the set of tangible markings TS implies that a transformation exists of the semi-Markov process associated with every GSPN system into a CTMC. The steady-state probability distribution over the tangible markings can thus be also obtained by a direct solution of this CTMC. In our case, the infinitesimal generator $\boldsymbol{Q}'$ of the CTMC associated with a GSPN can be constructed from the transition probability rate matrix $\boldsymbol{U}'$ of the REMC by dividing each of its rows by the mean sojourn time of the corresponding tangible marking. To conform with the standard definition of the infinitesimal generators, the diagonal elements of $\boldsymbol{Q}'$ are set equal to the negative sum of the off-diagonal components:

$$q'_{ij} = \begin{cases} \frac{1}{SJ_i}\,u'_{ij} & i \neq j \\ \\ -\sum_{j \neq i} q'_{ij} & i = j \end{cases} \tag{53}$$

This result shows that GSPNs, like SPNs, can be analysed by solving properly associated CTMCs. This obviously implies that the transient state probability distribution is the solution of the following differential equation

$$\frac{d\boldsymbol{\pi}'(\tau)}{d\tau} = \boldsymbol{\pi}'(\tau)\boldsymbol{Q}' \tag{54}$$

while the steady-state probability distribution over the tangible markings requires the solution of the following system of linear equations:

$$\begin{cases} \boldsymbol{\pi'}\,\boldsymbol{Q'} = \mathbf{0} \\ \\ \boldsymbol{\pi'}\,\mathbf{1}^{\mathrm{T}} = 1 \end{cases} \tag{55}$$

The computation of the performance indices defined over GSPN models can be performed using the reward method discussed in Section 7.1 without any additional difficulty.

The advantage of solving the system by first identifying the REMC is twofold. First, the time and space complexity of the solution is reduced in most cases, since the iterative methods used to solve the system of linear equations tend to converge more slowly when applied with sparse matrices and an improvement is obtained by eliminating the vanishing states thus obtaining a denser matrix [29,15]. Second, by decreasing the impact of the size of the set of vanishing states on the complexity of the solution method, we are allowed a greater freedom in the explicit specification of the logical conditions of the original GSPN, making it easier to understand.

The method outlined in this section exploits the elegant mathematical structure of the problem to overcome the difficulties due to the presence of loops of immediate transitions. The loops considered in this derivation are of the "transient" type [29] and correspond to situations in which a steady-state analysis of the model is possible. The REMC is instead impossible to construct following this approach when the loop of immediate transitions is of the "absorbing" type so that during its evolution the net can be trapped into a situation from which it cannot exit. Except for very pathological cases [29] in which the model makes sense despite the presence of such absorbing loops, GSPNs of this type are considered non-well behaving and their analysis is stopped once the existence of absorbing loops of immediate transitions is discovered during the construction of the infinitesimal generator of the REMC.

Computational Considerations - The mathematically elegant solution techniques outlined in the previous part of this section suffer in practice of the difficulties due to the size of the CTMCs associated with these models and of the time-scale differences of the activities represented by transitions. In this subsection we will discuss the main difficulties encountered for the computation of the transient and steady-state probability distributions and we will outline some solution techniques; we will refer to the specialized literature for a deeper discussion of the problem.

Transient Solution - Uniformization Method

The first difficulty in the analysis of the CTMC associated with GSPN models comes from the evaluation of the transient probability distribution on the markings of the net.

The apparently simple solution of Equation (8.5)

$$\boldsymbol{\pi}'(\tau) \;=\; \boldsymbol{\pi}'(0)e^{\boldsymbol{Q}'\tau} \;=\; \boldsymbol{\pi}'(0)\sum_{k=0}^{\infty}\frac{(\boldsymbol{Q}'\tau)^k}{k!} \qquad (56)$$

is unfortunately often rather difficult and unstable to compute [67]. Moler and Van Loan [19] discuss nineteen "dubious" ways to compute the exponential of relatively small matrices whose accuracy heavily depends on the norms of the matrices.

One of the most commonly used methods for computing the transient probabilities and that avoids most of these problems is called the *uniformization* technique that is based on the following simple derivation.

Assume that the diagonal elements of the infinitesimal generator $\boldsymbol{Q}'$ are bounded, so that there exist a Γ such that:

$$|q'_{ii}| \;\leq\; \Gamma \;<\; \infty, \;\; \forall \boldsymbol{m}_i \;\in\; TS \tag{57}$$

A (discretized) transition probability matrix $\boldsymbol{R}$ can be defined as follows

$$\boldsymbol{R} \;=\; \boldsymbol{I} \;+\; \frac{1}{\Gamma}\boldsymbol{Q}' \tag{58}$$

From this definition we have that $\boldsymbol{Q}' = \Gamma(\boldsymbol{R} - \boldsymbol{I})$, so that

$$\boldsymbol{\pi}'(\tau) \;=\; \boldsymbol{\pi}'(0)e^{\boldsymbol{Q}'\tau} \;=\; \boldsymbol{\pi}'(0)e^{\tau\Gamma\boldsymbol{R}-\tau\Gamma\boldsymbol{I}} \;=\; \boldsymbol{\pi}'(0)e^{-\tau\Gamma}e^{\tau\Gamma\boldsymbol{R}} \tag{59}$$

since $e^{-(\tau\Gamma)\boldsymbol{I}} = e^{-\tau\Gamma}\boldsymbol{I}$ and $\boldsymbol{R}$ and $\boldsymbol{I}$ commute. The transient distribution at time τ is thus obtained by computing an approximation to the infinite summation

$$\boldsymbol{\pi}'(\tau) \;=\; \boldsymbol{\pi}'(0)\,\sum_{k=0}^{\infty}\boldsymbol{R}^k e^{-\Gamma\tau}\frac{(\Gamma\tau)^k}{k!} \tag{60}$$

which is called the *uniformization equation* [67]. The advantages of this technique are the simplicity of its implementation and the possibility to control the approximation error. In fact, it is possible to easily identify the limit after which to truncate the series in order to reduce the resulting error below any predefined threshold ϵ [67]. The approximated result can thus be expressed in the following form:

$$\boldsymbol{\pi}'(\tau) \;=\; \boldsymbol{\pi}'(0)\,\sum_{k=0}^{K}\boldsymbol{R}^k e^{-\Gamma\tau}\frac{(\Gamma\tau)^k}{k!} \tag{61}$$

Steady-State Solution - Time Scale Decomposition

As we have seen at the beginning of this section, the steady state probability distribution of the markings of a GSPN model is obtained from the solution of a system of linear equations. This problem, that is mathematically simple and well understood, may pose considerable difficulties in the case of GSPNs due to the size of their reachability set and to the possibility of having within the infinitesimal generator of the corresponding CTMC rates that are several orders of magnitude different from each other. In this section we will not survey the many methods that can be employed for the solution of these large system of linear equations; the interested reader is referred to [67] for a comprehensive discussion of direct as well as of iterative techniques that can be employed to obtain the desired solution keeping under control the storage requirements as well as the computational costs. We will instead briefly outline a technique that can be conveniently used to obtain approximate results when the GSPN models are of a special type and when transitions of different speeds are included in the net.

Consider the case of the operation of a simple processor/memory system in which the memory may fail while it is not accessed. Fig. 28 depicts the GSPN model of such a system in which failures happen quite rarely and repairs require a considerable amount of time to be completed. In a model of this type, besides the distinction between immediate and timed transitions, we can further classify timed transitions as *fast* and *slow*. In Fig. 28, transitions T_{req} and T_{str} can be considered fast, while transitions T_{fail} and T_{rep} can be assumed to be slow.

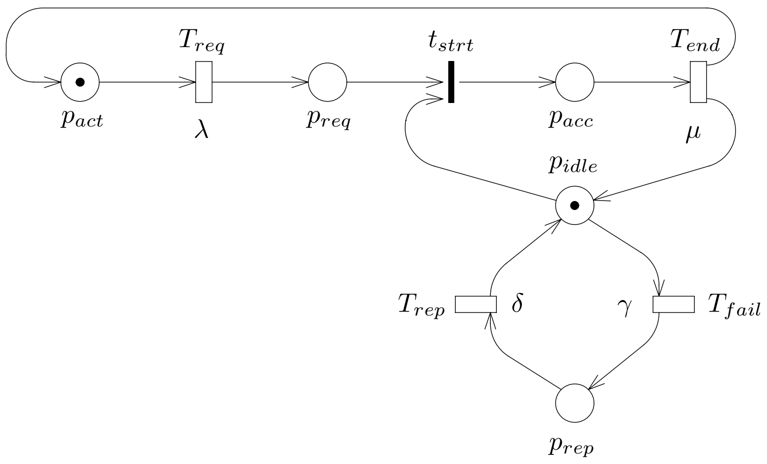


Fig. 28. A simple processor/memory system with failures

A technique for solving these models in a computationally convenient manner has been proposed by Ammar and Islam [11] and called Time Scale Decomposition Method (TSDM). This approach is based on the theory of *Near Complete Decomposability* due to Simon and Ando [66] and further investigated by Courtois [33]. In this case we may only observe that at the fast time scale, failures and repairs are so far apart in time, that during these intervals the system can be assumed to be fault-free. The theory of near decomposable systems says that as the process approaches the long-term equilibrium, a short-term equilibrium is reached between rare events, so that the short-term analysis can be approximately separated from the analysis of the long-run dynamics of such systems. The theory is developed by identifying within the Markov chains representing these systems, groups of states with (internal) strong interaction compared with the weak interaction existing among states of different groups. The method of

Ammar and Islam tries to identify these groups at the GSPN level by removing the slow transitions so that the net decomposes into several "fast" subnets that describe the short-run dynamics of the system. The individual solution of the CTMCs associated with these subnets provides the basis for the computation of aggregated representations that are used in the construction of a reduced subnet, called the aggregated GSPN (AGSPN), used for the analysis of the long-run dynamics of the system.

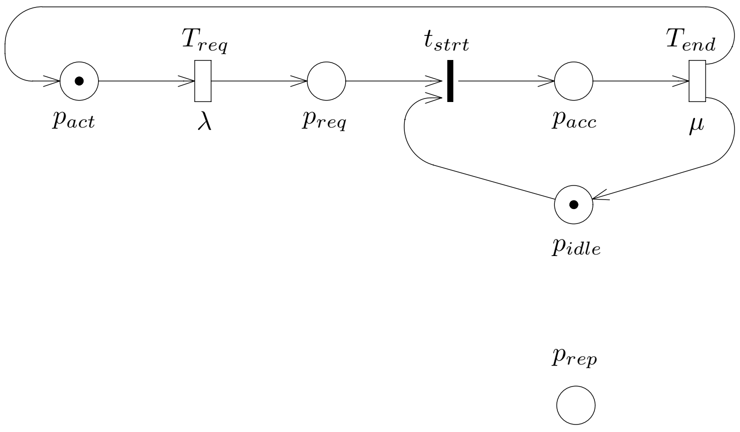


Fig. 29. Fast subnets derived form the simple processor/memory system with failures of Fig. 28

Following the technique proposed by Ammar and Islam, we remove the slow transitions from the net of Fig. 28 and we obtain the two subnets of Fig. 29, only one of which is interesting for the construction of the aggregated net of Fig. 30. Computing the solution of this first subnet (which is very simple since it has only two tangible markings corresponding to the processor active locally and to the processor accessing the memory), we observe that the firing rate of transition T_{a1} of the aggregated net is different from zero only when this fast submodels is in its first state.

Assuming that $\rho = \lambda/\mu$ and $\sigma = \gamma/\delta$, and indicating with $\boldsymbol{\pi}$ the solution of the original model of Fig. 28, with $\boldsymbol{\pi}^*$ its approximation computed with the TSDM of Ammar and Islam, with $\boldsymbol{\pi}^1$ the solution of the fast submodel of Fig. 29, and with P the solution of the aggregated model of Fig. 30, we have:

$$\pi^1(p_{act} + p_{idle}) \;=\; \frac{1}{1\;+\;\rho}. \qquad (62)$$

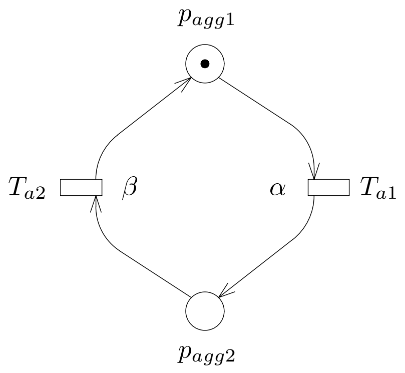


Fig. 30. Aggregated subnet derived form the simple processor/memory system with failures of Fig. 28

From the solution of the fast model, we have that the equivalent failure rate is:

$$\alpha \; = \; \frac{\gamma}{1 \; + \; \rho}, \tag{63}$$

while the repair rate remain unchanged:

$$\beta \; = \; \delta. \tag{64}$$

With these definitions, the solution of the aggregated (high level) model is easily obtained:

$$\mathcal{P}(p_{agg1}) \; = \; \frac{1 \; + \; \rho}{1 \; + \; \rho \; + \; \sigma}. \tag{65}$$

Once we know the probabilities of the aggregated states, we use them as conditional probabilities for obtaining detailed information on the states of the fast models. We thus obtain:

$$\pi^*(p_{act} + p_{idle}) \; = \; \frac{1}{1 \; + \; \rho \; + \; \sigma} \tag{66}$$

Solving the original (whole) model directly, we would have obtained:

$$\pi(p_{act} + p_{idle}) \; = \; \frac{1}{1 \; + \; \rho \; + \; \sigma \; + \; \rho\sigma\dfrac{\delta}{\lambda + \delta}}. \tag{67}$$

The difference between $\boldsymbol{\pi}$ and $\boldsymbol{\pi}^*$ is due to the approximate computation of the aggregated rate α of transition T_{a1} of the model of Fig. 29. The aggregated result would have been exact if the equivalent failure rate α had the following expression:

$$\alpha \; = \; \frac{\gamma}{1 \; + \; \rho \, (1 \; + \; \frac{\gamma}{\lambda+\delta})} \tag{68}$$

that differs from that of Equation (8.5) only for the ratio $\gamma/(\lambda+\delta)$ that obviously becomes negligible when $\gamma \ll \lambda$ as it is the case for our model.

In general the technique presented in [11] cannot be applied in such a straightforward manner and requires some clever intervention from the analyst, thus making it unsuitable for automatic applications. Under the condition of dealing with a somehow more restricted class of models, a variation of the above technique has been presented by Blakemore and Tripathi [16] that, working at the level of the state space of the CTMC underlying the GSPN model, first define the groups of states used for the aggregation, and then identify the transitions (of the GSPN model) that make the CTMC moving among these groups of states. Depending on the choice of the groups of states, the transitions that induce a change of state between groups (called *cross transitions*) can be either fast or slow. An implementation of this method should make sure that cross transitions are also slow in order for the TSD technique to be accurate, issuing a warning when this condition is not met. The interested reader will find the details of this last method, together with an algorithm for its automatic application, in [16].

8.6 Reducing GSPNs to SPNs

As we saw in the previous sections, immediate transitions, that have quite a beneficial impact on the modelling power of GSPNs, unfortunately make the overall analysis of GSPNs more complex than that of SPNs. Indeed, while for the latter the reachability graph is isomorphic to the state transition rate diagram of the CTMC underlying the SPN, such isomorphism does not exist for GSPNs, but is obtained only after the elimination of vanishing markings. Since loops of immediate transitions seldom appear in GSPN models, several solution methods have been devised that discard the vanishing states "on the fly" thus trading space with computational effort due to the fact that in some situations sequences of vanishing states are repeatedly generated to compute the transition rates among different pairs of tangible markings. Usually this method is computationally convenient, even if some extra work must be performed to preanalyze the model in order to discover the presence of situations that can not be treated with this technique. The choice of not saving vanishing markings has also the drawback of making certain performance indices related with the firing of immediate transitions impossible to compute. Details on the comparison of the advantages and disadvantages of storing vanishing markings can be found in [29]. An analysis procedure different from that described in the previous sections consists in the elimination of all immediate transitions from a given GSPN before starting the generation of its reachability set, so as to produce an SPN whose underlying continuous-time Markov chain is identical to that corresponding to the GSPN.

A complete and formal description of these reduction rules and of the class of GSPNs for which it is possible to compute an equivalent SPN, can be found in [22,34]. Here we only provide a concise and informal overview of the rules for the reduction of a GSPN model to an equivalent SPN in the case of free-choice conflicts.

The basic idea behind the elimination of immediate transitions is quite simple and can be easily explained in its natural form by means of an example. The model depicted in Fig. 31(a) is a free-choice GSPN subnet whose SPN counterpart is shown in Fig. 31(b). The two systems are equivalent as far as tangible states are concerned.

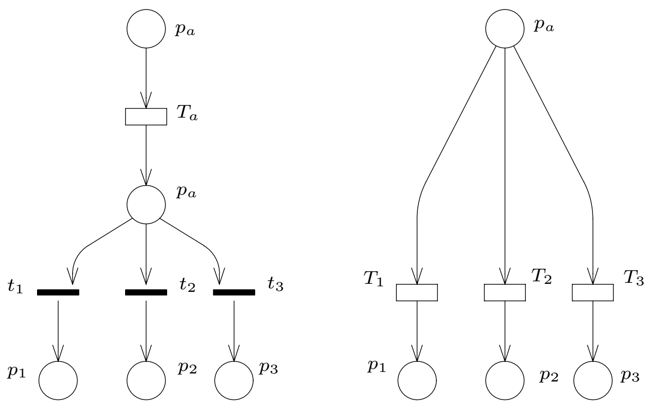


Fig. 31. Removing immediate transitions: the free-choice case

To understand how we can transform the GSPN into its corresponding SPN, consider what happens in the GSPN when T_a fires: the three immediate transitions t_1, t_2, and t_3 become enabled, due to the arrival of one token in place p_b; they are the only transitions that are enabled in the subnet, and the choice about which one to fire is made according to their associated weights. The basic behaviour of the subnet in Fig. 31(a) is therefore that of "moving" tokens from place p_a to one of the three places p_1, p_2, and p_3. The net in Fig. 31(b) is clearly equivalent to that of Fig. 31(a) from the point of view of the flow of tokens (token flow equivalence was defined in [13]).

When time is involved, token flow equivalence is not enough to ensure the possibility of freely interchanging the two types of models. In particular, the equivalence notion we are interested in must preserve the underlying stochastic behaviour of the net. We must therefore take into account not only the possible final states of the subnet, but also the rates at which the model transits from state to state.

The operations of the GSPN subnet induce a movement of a token from p_a to p_k, $k = 1, 2, 3$, with rate

$$\frac{w_a w_k}{\sum_{j=1,2,3} w_j} \tag{69}$$

where w_a is the rate of the exponential distribution associated with timed transition T_a, and w_k is the weight of immediate transition t_k, so that

$$\frac{w_k}{\sum_{j=1,2,3} w_j} \tag{70}$$

is the probability that t_k is the transition that actually fires when t_1, t_2, and t_3 are all enabled.

The timed transition T_k in the SPN model represents the firing of timed transition T_a followed by immediate transition t_k. If we define its firing rate as in Equation (69), then the rate at which the SPN subnet in Fig. 31(b) induces a movement of a token from p_a to p_k, given that transition T_a is enabled, is exactly the same as for the GSPN subnet, and therefore the rates of the underlying Markov processes are the same. Note that place p_b was deleted in the reduction process: this is not surprising because p_b is a vanishing place, a place that is never marked in a tangible marking.

The elimination procedure is somewhat more complex if the set of immediate transitions enabled by the firing of a timed transition does not form a free-choice conflict. Indeed, in this case the probability of firing an immediate transition may depend also on the other simultaneously enabled immediate transitions. We are thus forced to consider all possible cases. Details on this elimination procedure can be found in [22,34].

9 Conclusions

In this paper we have shown how Stochastic Petri Nets can be conveniently used for the analysis of complex models of DEDS and for their performance and reliability evaluation.

The advantage of net-based models, however, goes far beyond the modelling power of the formalism. In fact, the analysis of the structure of the graph and the computation of its algebraic properties provide information such as invariant conditions, flow-balance equations, and special structures of the reachability graph that can be used to identify models with peculiar solution characteristics, to optimize the solution techniques, and to develop approximation methods.

After an initial difficulty in accepting this new formalism due to its intrinsic complexity, SPNs are widely used today for the performance and reliability evaluation of many practical systems.

What was originally perceived as a disadvantage is now often appreciated because of the capability of the formalism to describe with precision complex phenomena that are typical of distributed and parallel systems. Moreover, the possibility of using the same model for both a qualitative analysis (functional validation) and an efficiency and reliability evaluation is understood as an important result in the assessment of real systems. It is a common belief that this approach should be further exploited and that more powerful and more user-friendly software tools must be developed to make this integrated study common practice for system designers.

This success also highlights a whole set of new problems since larger and larger models are being built and need to be analyzed. Dealing with large models is obviously difficult since even in the case of bounded nets, the size of their reachability sets can become enormous, making their numerical evaluation impossible and discrete-event simulation extremely expensive.

Many important results have been developed in the SPN field since the time of the introduction of this formalism. In our view, the most important ones are those using the net structure of the model to ease the modelling effort and to improve the efficiency of the solution methods. This allows the analyst to reason about the system at the net level while hiding the complexity of the underlying probabilistic structure.

We can summarize these new developments as follows:

– *General Distributions* - The assumption of the negative exponential distribution of firing times was soon felt to be too restrictive for a convenient representation of complex systems and several attempts were made to introduce general firing time distributions into the formalism. Firing delays that are characterized by phase-type distributions [57], were introduced by employing suitable subnet structures that could be easily embedded into GSPN models [20].
  Using the approach of identifying an embedded Markov chain, a technique has been proposed for the analysis of deterministic and stochastic Petri nets (DSPNs) [8]. In this case the embedded chain is used for the computation of the steady-state solution of nets in which at most one transition of constant delay is enabled in any marking. The transient analysis of DSPN models is presented in [27].
  It was soon recognized that when allowing full generality in the specification of the model the possibility for the computation of analytical or numerical solutions was lost and simulation was the only way to analyze these models [41,40].
  Recently, the class of DSPN models has been extended allowing the transitions to have generally distributed firing times, provided that the constraint of having at most one of these transitions enabled in each marking is still satisfied [39]. This class of models is also called Markov Regenerative SPNs [48,28] and special formulas for the computation of their steady-state solution were derived. A systematic study of this class of models can be found in [30].
– *Symmetries* - When dealing with complex systems, it often happens that their models can be constructed via the replication of many identical submodels. To deal with this problem, coloured Petri nets have been proposed to allow the construction of more compact representations [47].
  A special class of coloured Petri nets is that of *Stochastic Well Formed Petri Nets* (SWNs) in which restrictions are introduced on the functions that regulate transition firings and colour manipulations [36,25,24]. The important feature of SWNs is that the special form of their colour functions allows the direct construction of an aggregated state space. With this formalism the

symmetries intrinsic in the model are directly exploited to identify markings that are representative of large groups of states having similar characteristics. The aggregation method is fully automated and the direct generation of the aggregated Markov chain is obtained with considerable saving at the level of memory requirements. A significant advantage is also obtained when the reduced state space is still too large for the model to be solved numerically and we must resort to simulation to obtain the performance indices of interest. A method of symbolic simulation has been developed in which only the aggregated markings are generated [26].

– *Block Structure* - The problem of mastering the complexity of models of real systems has also been investigated using an approach driven by the idea that efficient solutions of difficult problems must be sought from the earliest stage of model construction. This approach tries to exploit the properties and the structure of the net to guide the solution method. This principle implies that, in order to understand how complex systems work, their behaviours must be considered as following from the composition of their individual parts. Thus in the development of the model, the analyst must maintain a local view of the different components avoiding the direct representation of global system features that, resulting from the interaction of the individual parts, could exhibit unexpected behaviours in pathological cases that are difficult to foresee. The Markov chain that underlies the entire model is only formally specified in this approach in terms of the Markov chains (transition probability matrices) of the individual submodels and of certain correcting factors that account for the interactions among the submodels. The complete transition probability matrix is never really constructed thereby allowing the solution of extremely large models in a very efficient manner. This solution technique also has the non-trivial advantage of being quite suitable for parallelization. In addition, the solution of the entire model is made easier when the submodels interact in very special ways [18], thus making the correcting factors extremely simple.
– *Product-Form SPNs* - To overcome the state-space explosion problem of the Markov chains associated with these models, a class of SPNs has been identified in which the steady-state probability distribution of their markings can be expressed in a product-form. The characterization of this class of SPNs was first expressed in terms of the special repetitive structures exhibited by their reachability graph [46,64] and subsequently in terms of structural criteria that can be easily checked by inspecting the incidence matrix of the net [43,42]. Proofs have been developed to show that it is possible to recognize whether SPNs have a product-form solution strictly from the results of their structural analysis. A complete characterization of this class of models can be found in [17].

Despite these results that we have briefly overviewed, the state-space explosion problem remains the major difficulty for using Petri net models in practical applications. There is general consensus that the only means of successfully dealing with large models is to use a “divide and conquer” approach in which the solution of the entire model is constructed on the basis of the solutions of its individual submodels.

Compositionality is a property that is not natural in the Petri net field, but the tendency of constructing complex models from many small "building blocks" is becoming increasingly common and a need exists for also developing this approach for Petri nets. New theoretical results, mostly inspired by the research on "process algebras" [52] and "box calculus" [14], are being derived in this direction contributing to the development of a solid theoretical framework for compositionality both at the level of model construction and model solution.

Acknowledgement - The discussion contained in this work builds on the contents of several papers that the author has previously written on this topics in collaboration with several colleagues. It is thanks to them the extent to which the material contained in this report provides a good introduction to the use of Stochastic Petri Nets; any omissions and imprecisions, which are sure to exist, are instead the sole responsibility of the author. Special thanks go to Simona Bernardi for her help in harmonizing the notations and the figures throughout the whole report.

References

1. T. Agerwala. Putting Petri nets to work. *IEEE Computer*, pages 85–94, December 1979.
2. M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, and A. Cumani. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, 15(7):832–846, July 1989.
3. M. Ajmone Marsan, G. Balbo, G. Chiola, and G. Conte. Generalized stochastic Petri nets revisited: Random switches and priorities. In *Proc. Intern. Workshop on Petri Nets and Performance Models*, pages 44–53, Madison, WI, USA, August 1987. IEEE-CS Press.
4. M. Ajmone Marsan, G. Balbo, and G. Conte. A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(1), May 1984.
5. M. Ajmone Marsan, G. Balbo, and G. Conte. *Performance Models of Multiprocessor Systems.* MIT Press, Cambridge, USA, 1986.
6. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* J. Wiley, 1995.
7. M. Ajmone Marsan, G. Balbo, and K.S. Trivedi, editors. *Proc. Intern. Workshop on Timed Petri Nets*, Torino, Italy, July 1985. IEEE-CS Press.
8. M. Ajmone Marsan and G. Chiola. On Petri nets with deterministic and exponential transition firing times. In *Proc. 7^{th} European Workshop on Application and Theory of Petri Nets*, pages 151–165, Oxford, England, June 1986.
9. H. Alaiwan and G. Memmi. Algorithmes de recherche des solutions entieres positives d'un systeme d'equations lineaires homogeneus en nombres entieres. *Revue Technique Thomson-CSF*, 14(1):125–135, March 1982. in French.
10. H. Alaiwan and J. M. Toudic. Research des semiflots, des verrous et des trappes dans le reseaux de Petri. *Technique et Science Informatiques*, 4(1), February 1985.

11. H.H. Ammar and S.M. Rezaul Islam. Time scale decomposition of a class of generalized stochastic Petri net models. *IEEE Transactions on Software Engineering*, 15(6):809–820, June 1989.
12. G. Balbo, M. Silva, and editors. *Performance Models for Discrete Event Systems with Synchronizations: Formalisms and Analysis Techniques.* KRONOS, Zaragoza, Spain, 1998.
13. G. Berthelot. Transformations and decompositions of nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets '86 - Part I*, volume 254 of *LNCS*, pages 359–376. Springer Verlag, Bad Honnef, West Germany, February 1987.
14. E. Best, R. Devillers, and J. Hall. The Petri box calculus: a new causal algebra with multilabel communication. In G. Rozenberg, editor, *Advances in Petri Nets*, volume 609 of *LNCS*, pages 21–69. Springer Verlag, 1992.
15. A. Blakemore. The cost of eliminating vanishing markings from generalized stochastic Petri nets. In *Proc. 3rd Intern. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989. IEEE-CS Press.
16. A. Blakemore and S.K. Tripathi. Automated time scale decomposition and analysis of stochastic Petri nets. In *Proc. 5th Intern. Workshop on Petri Nets and Performance Models*, pages 248–257, Toulouse, France, October 1993. IEEE-CS Press.
17. R.J. Boucherie. A characterization of independence for competing Markov chains with applications to stochastic Petri nets. In *Proc. 5th Intern. Workshop on Petri Nets and Performance Models*, pages 117–126, Toulouse, France, October 1993. IEEE-CS Press.
18. P. Buchholz. Hierarchical high level Petri nets for complex system analysis. In *Proc. 15th Intern. Conference on Applications and Theory of Petri Nets*, number 185 in LNCS, Zaragoza, Spain, 1994. Springer-Verlag.
19. Moler C. and C. Van Loan. Nineteen doubious ways to compute the exponential of a matrix. *SIAM Review*, 20(4):801–836, 1978.
20. P. Chen, S.C. Bruell, and G. Balbo. Alternative methods for incorporating non-exponential distributions into stochastic Petri nets. In *Proc. 3rd Intern. Workshop on Petri Nets and Performance Models*, pages 187–197, Kyoto, Japan, December 1989. IEEE-CS Press.
21. G. Chiola, M. Ajmone Marsan, G. Balbo, and G. Conte. Generalized stochastic Petri nets: A definition at the net level and its implications. *IEEE Transactions on Software Engineering*, 19(2):89–107, February 1993.
22. G. Chiola, S. Donatelli, and G. Franceschinis. GSPN versus SPN: what is the actual role of immediate transitions? In *Proc. 4th Intern. Workshop on Petri Nets and Performance Models*, pages 20–31, Melbourne, Australia, December 1991. IEEE-CS Press.
23. G. Chiola, S. Donatelli, and G. Franceschinis. Priorities, inhibitor arcs, and concurrency in P/T nets. In *Proc. 12th Intern. Conference on Application and Theory of Petri Nets*, Aarhus, Denmark, June 1991.
24. G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic Well-Formed coloured nets for symmetric modelling applications. *IEEE Transactions on Computers*, 42(11), November 1993.
25. G. Chiola and G. Franceschinis. Colored GSPN models and automatic symmetry detection. In *Proc. 3rd Intern. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989. IEEE-CS Press.

26. G. Chiola, G. Franceschinis, R. Gaeta, and M. Ribaudo. GreatSPN1.7: GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. *Performance Evaluation*, 24, 1995. Special Issues on Performance Modelling Tools.
27. H. Choi, G. Kulkarni, and K.S. Trivedi. Transient analysis of deterministic and stochastic petri nets. In *Proc. 14^{th} Intern. Conference on Application and Theory of Petri Nets*, Chicago, Illinois, June 1993. Springer Verlag.
28. H. Choi, V.S. Kulkarni, and K.S. Trivedi. Markov regenerative stochastic Petri nets. In *Proc. of Performance 93*, Rome, Italy, September 1993.
29. G. Ciardo. *Analysis of large Petri net models.* PhD thesis, Department of Computer Science, Duke University, Durham, NC, USA, 1989. Ph. D. Thesis.
30. G. Ciardo, R. German, and C. Lindemann. A characterization of the stochastic process underlying a stochastic Petri net. In *Proc. 5^{th} Intern. Workshop on Petri Nets and Performance Models*, pages 170–179, Toulouse, France, October 1993. IEEE-CS Press.
31. G. Ciardo and C. Lindemann. Analysis of deterministic and stochastic Petri nets. In *Proc. 5^{th} Intern. Workshop on Petri Nets and Performance Models*, pages 160–169, Toulouse, France, October 1993. IEEE-CS Press.
32. J.E. Coolhan and N. Roussopoulos. Timing requirements for time-driven systems using augmented Petri nets. In *Proc. Intern. Workshop on Timed Petri Nets*, Torino, Italy, July 1985. IEEE-CS Press.
33. P.J. Courtois. *Decomposability: Queueing and Computer Systems Applications.* Academic Press, New York, 1977.
34. S. Donatelli. *L'uso delle reti di Petri per la valutazione e la validazione di sistemi di grandi dimensioni.* PhD thesis, Dipartimento di Informatica, Università di Torino, Corso Svizzera 185, 10149, Torino, Italy, February 1990. (in italian).
35. J.B. Dugan, K.S. Trivedi, R.M. Geist, and V.F. Nicola. Extended stochastic Petri nets: Applications and analysis. In *Proc. PERFORMANCE '84*, Paris, France, December 1984.
36. C. Dutheillet and S. Haddad. Aggregation and disaggregation of states in colored stochastic Petri nets: Application to a multiprocessor architecture. In *Proc. 3^{rd} Intern. Workshop on Petri Nets and Performance Models*, Kyoto, Japan, December 1989. IEEE-CS Press.
37. G. Florin and S. Natkin. Les reseaux de Petri stochastiques. *Technique et Science Informatiques*, 4(1), February 1985.
38. G. Florin and S. Natkin. Matrix product form solution for closed synchronized queueing networks. In *Proc. 3^{rd} Intern. Workshop on Petri Nets and Performance Models*, pages 29–39, Kyoto, Japan, December 1989. IEEE-CS Press.
39. R. German and C. Lindemann. Analysis of stochastic Petri nets by the method of supplementary variables. In *Proc. of Performance 93*, Rome, Italy, September 1993.
40. P. J. Haas and G. S. Shedler. Regenerative stochastic Petri nets. *Performance Evaluation*, 6(3):189–204, September 1986.
41. P.J. Haas and G.S. Shedler. Regenerative simulation of stochastic Petri nets. In *Proc. Intern. Workshop on Timed Petri Nets*, Torino, Italy, July 1985. IEEE-CS Press.
42. W. Henderson and D. Lucic. Exact results in the aggregation and disaggregation of stochastic Petri nets. In *Proc. 4^{th} Intern. Workshop on Petri Nets and Performance Models*, pages 166–175, Melbourne, Australia, December 1991. IEEE-CS Press.
43. W. Henderson and P.G. Taylor. Aggregation methods in exact performance analysis of stochastic Petri nets. In *Proc. 3^{rd} Intern. Workshop on Petri Nets and Performance Models*, pages 12–18, Kyoto, Japan, December 1989. IEEE-CS Press.

44. M.A. Holliday and M.K. Vernon. A generalized timed Petri net model for performance analysis. In *Proc. Intern. Workshop on Timed Petri Nets*, Torino, Italy, July 1985. IEEE-CS Press.
45. K. Lautenbach. Linear algebraic technique for place/transition nets. In W. Brawer, W. Reisig, and G. Rozenberg, editors, *Advances on Petri Nets '86 - Part I*, volume 254 of *LNCS*, pages 142–167. Springer Verlag, Bad Honnef, West Germany, February 1987.
46. A.A. Lazar and T.G. Robertazzi. Markovian Petri net protocols with product form solution. *Performance Evaluation*, 12:67–77, 1991.
47. C. Lin and D.C. Marinescu. On stochastic high level Petri nets. In *Proc. Intern. Workshop on Petri Nets and Performance Models*, Madison, WI, USA, August 1987. IEEE-CS Press.
48. V. Mainkar, H. Choi, and K. Trivedi. Sensitivity analysis of Markov regenerative stochastic Petri nets. In *Proc.* 5^{th} *Intern. Workshop on Petri Nets and Performance Models*, Toulouse, France, October 1993. IEEE-CS Press.
49. J. Martinez and M. Silva. A simple and fast algorithm to obtain all invariants of a generalized Petri net. In *Proc.* 2^{nd} *European Workshop on Application and Theory of Petri Nets*, Bad Honnef, West Germany, September 1981. Springer Verlag.
50. P. M. Merlin and D. J. Farber. Recoverability of communication protocols: Implications of a theoretical study. *IEEE Transactions on Communications*, 24(9):1036–1043, September 1976.
51. J. F. Meyer, A. Movaghar, and W. H. Sanders. Stochastic activity networks: Structure, behavior, and application. In *Proc. Intern. Workshop on Timed Petri Nets*, pages 106–115, Torino,Italy, July 1985.
52. R. Milner. *Communication and concurrency.* Prentice Hall, 1989.
53. M.K. Molloy. *On the Integration of Delay and Throughput Measures in Distributed Processing Models.* PhD thesis, UCLA, Los Angeles, CA, 1981. Ph.D. Thesis.
54. M.K. Molloy, T. Murata, and M.K. Vernon, editors. *Proc. Intern. Workshop on Petri Nets and Performance Models*, Madison, Wisconsin, August 1987. IEEE-CS Press.
55. T. Murata. Petri nets: properties, analysis, and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
56. S. Natkin. *Les Reseaux de Petri Stochastiques et leur Application a l'Evaluation des Systemes Informatiques.* PhD thesis, CNAM, Paris, France, 1980. These de Docteur Ingegneur.
57. M.F. Neuts. *Matrix Geometric Solutions in Stochastic Models.* Johns Hopkins University Press, Baltimore, MD, 1981.
58. J. D. Noe and G. J. Nutt. Macro e-nets representation of parallel systems. *IEEE Transactions on Computers*, 31(9):718–727, August 1973.
59. J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.
60. C.A. Petri. Communication with automata. Technical Report RADC-TR-65-377, Rome Air Dev. Center, New York, NY, 1966.
61. C. V. Ramamoorthy and G. S. Ho. Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE Transactions on Software Engineering*, 6(5):440–449, September 1980.
62. C. Ramchandani. *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets.* PhD thesis, MIT, Cambridge, MA, 1974. Ph.D. Thesis.
63. W. Reisig. *Petri Nets: an Introduction.* Springer Verlag, 1985.
64. T.G. Robertazzi. *Computer Networks and Systems: Queueing Theory and Performance Evaluation.* Springer Verlag, 1991.

65. J. Sifakis. Petri nets for performance evaluation. In H. Beilner and E. Gelenbe, editors, *Proc. 3^{rd} Intern. Symp. IFIP*, pages 75–93, 1978.
66. H.A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29, 1961.
67. W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains.* Princeton University Press, Princeton, New Jersey, USA, 1994.
68. F. J. W. Symons. Introduction to numerical Petri nets, a general graphical model of concurrent processing systems. *Australian Telecommunications Research*, 14(1):28–33, January 1980.
69. R.S. Varga. *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1962.
70. C.Y. Wong, T.S. Dillon, and K.E. Forward. Timed places Petri nets with stochastic representation of place time. In *Proc. Intern. Workshop on Timed Petri Nets*, Torino, Italy, July 1985. IEEE-CS Press.
71. W.M. Zuberek. Timed Petri nets and preliminary performance evaluation. In *Proc. 7^{th} Annual Symposium on Computer Architecture*, pages 88–96, La Baule, France, May 1980.